

2114

PTO/SB/21 (09-04)

Approved for use through 07/31/2006. OMB 0651-0031

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

## TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Total Number of Pages in This Submission

Application Number	09/752,837
Filing Date	12/28/2000
First Named Inventor	Nobuyuki YAMAUCHI et al.
Art Unit	2114
Examiner Name	Dieu Minh T. LE
Attorney Docket Number	44471/251413

### ENCLOSURES (Check all that apply)

- |  |  |   |
|--|--|---|
| <input type="checkbox"/> Fee Transmittal Form<br><input type="checkbox"/> Fee Attached<br><input type="checkbox"/> Amendment/Reply<br><input type="checkbox"/> After Final<br><input type="checkbox"/> Affidavits/declaration(s)<br><input type="checkbox"/> Extension of Time Request<br><input type="checkbox"/> Express Abandonment Request<br><input type="checkbox"/> Information Disclosure Statement<br><br><input type="checkbox"/> Certified Copy of Priority Document(s)<br><input type="checkbox"/> Reply to Missing Parts/<br>Incomplete Application<br><input type="checkbox"/> Reply to Missing Parts<br>under 37 CFR 1.52 or 1.53 | <input type="checkbox"/> Drawing(s)<br><input type="checkbox"/> Licensing-related Papers<br><br><input type="checkbox"/> Petition<br><input type="checkbox"/> Petition to Convert to a<br>Provisional Application<br><input type="checkbox"/> Power of Attorney, Revocation<br>Change of Correspondence Address<br><br><input type="checkbox"/> Terminal Disclaimer<br><input type="checkbox"/> Request for Refund<br><br><input type="checkbox"/> CD, Number of CD(s) _____<br><input type="checkbox"/> Landscape Table on CD | <input type="checkbox"/> After Allowance Communication to TC<br><br><input type="checkbox"/> Appeal Communication to Board<br>of Appeals and Interferences<br><br><input type="checkbox"/> Appeal Communication to TC<br>(Appeal Notice, Brief, Reply Brief)<br><br><input type="checkbox"/> Proprietary Information<br><br><input type="checkbox"/> Status Letter<br><input checked="" type="checkbox"/> Other Enclosure(s) (please identify<br>below):<br><br>Transmittal of Certified Copies<br>of Priority Documents JP<br>11-375859 and JP 2000-396112 |
|--|--|---|

Remarks

### SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	KILPATRICK STOCKTON LLP		
Signature			
Printed name	Brenda O. Holmes		
Date	11.04.2004	Reg. No.	40,339

### CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as certified first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below:

Signature			
Typed or printed name	Janie Wilkins	Date	11/4/2004

This collection of information is required by 37 CFR 1.5. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.



Patents

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Nobuyuki YAMAUCHI et al.

Application No. 09/752,837

Filed: December 28, 2000

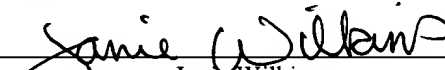
For: **Information Processing Apparatus  
Defect Analysis Program, Defect  
Analysis Method, and Application  
Program Development Assistance  
System**

Art Unit: 2114

Examiner: Le, Dieu Minh T.

Attorney Docket No.: 44471/251413

I hereby certify that this correspondence is being deposited with the United States Postal Service as certified first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on 11/04/04.

  
Jamie Wilkins

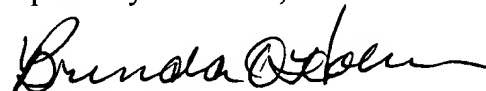
**TRANSMITTAL OF CERTIFIED COPIES OF PRIORITY DOCUMENTS**

Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Sir:

Submitted herewith are certified copies of the priority documents for the referenced patent applications: JP2000-396112 and JP11-375859.

Respectfully submitted,



Brenda O. Holmes  
Reg. No. 40,339

Kilpatrick Stockton LLP  
1100 Peachtree Street, Suite 2800  
Atlanta, Georgia 30309  
(404) 815-6500  
KS File: 44471/251413

S-643

日 本 国 特 許 庁  
PATENT OFFICE  
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日  
Date of Application:

2000年12月26日

出 願 番 号  
Application Number:

特願2000-396112

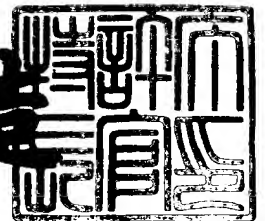
願 人  
Applicant(s):

東芝エルエスアイシステムサポート株式会社  
株式会社東芝

2001年 2月 9日

特許庁長官  
Commissioner,  
Patent Office

及 川 耕 造



CERTIFIED COPY OF  
PRIORITY DOCUMENT

出証番号 出証特2001-3006357

BEST AVAILABLE COPY

PATENT OFFICE  
JAPANESE GOVERNMENT

This is to certify that the annexed is a true copy of the following application  
as filed with this Office.

Date of Application : December 26, 2000

Application Number : P2000-396112

Applicant(s): TOSHIBA LSI SYSTEM SUPPORT KABUSHIKI KAISHA  
KABUSHIKI KAISHA TOSHIBA

February 9, 2001

Commissioner,  
Patent Office                      Kouzou OIKAWA

Number of Certificate: 2001-3006357

【書類名】 特許願

【整理番号】 46B007087

【提出日】 平成12年12月26日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 11/32

【発明の名称】 情報処理装置、不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体、不具合解析方法、及びアプリケーションプログラム開発支援システム

【請求項の数】 20

【発明者】

    【住所又は居所】 神奈川県川崎市幸区小向東芝町 1 番地 株式会社東芝  
                                マイクロエレクトロニクスセンター内

    【氏名】 澤岡 明

【発明者】

    【住所又は居所】 神奈川県川崎市幸区小向東芝町 1 番地 株式会社東芝  
                                マイクロエレクトロニクスセンター内

    【氏名】 山内 信之

【発明者】

    【住所又は居所】 神奈川県川崎市幸区堀川町 5 8 0 番地 東芝エルエスアイシステムサポート株式会社内

    【氏名】 松本 奈津美

【特許出願人】

    【識別番号】 598010562

    【氏名又は名称】 東芝エルエスアイシステムサポート株式会社

【特許出願人】

    【識別番号】 000003078

    【氏名又は名称】 株式会社 東芝

【代理人】

    【識別番号】 100083806

【弁理士】

【氏名又は名称】 三好 秀和

【電話番号】 03-3504-3075

【選任した代理人】

【識別番号】 100068342

【弁理士】

【氏名又は名称】 三好 保男

【選任した代理人】

【識別番号】 100100712

【弁理士】

【氏名又は名称】 岩▲崎▼ 幸邦

【選任した代理人】

【識別番号】 100100929

【弁理士】

【氏名又は名称】 川又 澄雄

【選任した代理人】

【識別番号】 100108707

【弁理士】

【氏名又は名称】 中村 友之

【選任した代理人】

【識別番号】 100095500

【弁理士】

【氏名又は名称】 伊藤 正和

【選任した代理人】

【識別番号】 100101247

【弁理士】

【氏名又は名称】 高橋 俊一

【選任した代理人】

【識別番号】 100098327

【弁理士】

【氏名又は名称】 高松 俊雄

【先の出願に基づく優先権主張】

【出願番号】 平成11年特許願第375859号

【出願日】 平成11年12月28日

【手数料の表示】

【予納台帳番号】 001982

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9707392

【包括委任状番号】 9807824

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 情報処理装置、不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体、不具合解析方法、及びアプリケーションプログラム開発支援システム

【特許請求の範囲】

【請求項 1】 プログラムの実行履歴情報をもとに、該プログラムの動作状況を時系列にユーザーに対して表示し、表示された動作状況の中でユーザーが指定する不具合の箇所を受付けるユーザーインタフェース手段と、

前記ユーザーインタフェース手段によってユーザーから指定された不具合の箇所と該プログラムの動作状況から不具合要因を解析し、この不具合要因を解決するための解決策を特定する動作解析手段とを有し、

前記動作解析手段は、前記解決策を反映させた動作状況を再作成し、

前記ユーザーインタフェース手段は、前記不具合要因と前記解決策と再作成した動作状況とをユーザーに表示することを特徴とする情報処理装置。

【請求項 2】 前記動作解析手段において、前記解決策を反映させる箇所を特定することができない場合に、

更に、ユーザーに該解決策を反映させる箇所の指定を促し、

前記動作解析手段は、このユーザーにより指定された箇所に該解決策を反映させた動作状況を再作成し、

前記ユーザーインタフェース手段は、前記不具合要因と解決策と再作成した動作状況とをユーザーに表示することを特徴とする請求項 1 記載の情報処理装置。

【請求項 3】 前記情報処理装置は、前記解決策を特定するためのプログラム機能対応情報を有することを特徴とする請求項 1 記載の情報処理装置。

【請求項 4】 前記ユーザーインタフェース手段は、ユーザーが不具合の箇所を座標位置により指定することで、該指定箇所を一意に認識することを特徴とする請求項 1 記載の情報処理装置。

【請求項 5】 前記情報処理装置は、更に、

前記解決策と前記実行履歴情報とをもとに、本来の仕様を満たすプログラムのスケルトンを自動生成することを特徴とする請求項 1 記載の情報処理装置。



【請求項 6】 プログラムの実行履歴情報をもとに、該プログラムの動作状況を時系列にユーザーに表示するステップと、

ユーザーから指定された不具合の箇所と該プログラムの動作状況から不具合要因を解析し、この不具合要因を解決するための解決策を特定するステップと、

前記解決策を反映させた動作状況を再作成するステップと、

前記不具合要因と前記解決策と再作成した動作状況とをユーザーに表示するステップと

をコンピュータに実現させることを特徴とする不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体。

【請求項 7】 請求項 6 に記載の不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体は、さらに、

前記解決策を反映させる箇所を特定することができない場合に、ユーザーに該解決策を反映させる箇所の指定を促すステップと、

この指定された箇所に該解決策を反映させた動作状況を再作成するステップと、

前記不具合要因と前記解決策と再作成した動作状況とをユーザーに表示するステップと

をコンピュータに実現させることを特徴とする不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体。

【請求項 8】 請求項 6 に記載の不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体は、

プログラム機能対応情報テーブルを用いて前記解決策を特定する機能をコンピュータに実現させることを特徴とする不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体。

【請求項 9】 請求項 6 に記載の不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体は、さらに、

前記解決策と前記実行履歴情報とをもとに、本来の仕様を満たすプログラムのスケルトンを自動生成するステップをコンピュータに実現させることを特徴とする不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体。

【請求項 1 0】 プログラムの実行履歴情報をもとに、該プログラムの動作状況を時系列にユーザーに表示するステップと、

ユーザーから指定された不具合の箇所と該プログラムの動作状況から不具合要因を解析し、この不具合要因を解決するための解決策を特定するステップと、

前記解決策を反映させた動作状況を再作成するステップと、

前記不具合要因と前記解決策と再作成した動作状況とをユーザーに表示するステップと

を含むことを特徴とする不具合解析方法。

【請求項 1 1】 請求項 1 0 記載の不具合解析方法は、さらに、

前記解決策を反映させる箇所を特定することができない場合に、ユーザーに該解決策を反映させる箇所の指定を促すステップと、

この指定された箇所に該解決策を反映させた動作状況を再作成するステップと

、

前記不具合要因と前記解決策と再作成した動作状況とをユーザーに表示するステップと

を含むことを特徴とする不具合解析方法。

【請求項 1 2】 請求項 1 0 記載の不具合解析方法は、

プログラム機能対応情報テーブルを用いて前記解決策を特定することを特徴とする不具合解析方法。

【請求項 1 3】 請求項 1 0 記載の不具合解析方法は、更に、

前記解決策と前記実行履歴情報とをもとに、本来の仕様を満たすプログラムのスケルトンを自動生成するステップを含むことを特徴とする不具合解析方法。

【請求項 1 4】 ハードウェア資源上で実行されるアプリケーションプログラムの開発に用いられる開発支援システムにおいて、

前記ハードウェア資源、およびソフトウェアを含む実行環境での実行時におけるシステム環境と、該実行環境の動作規則を定義する環境定義部と、

環境定義に基づいて前記実行環境におけるアプリケーションプログラムの仮想実行状態を検証するチェック部と、

前記仮想実行状態を可視的に表示するための表示情報を作成する表示情報作成

部と

を有することを特徴とするアプリケーションプログラム開発支援システム。

【請求項 1 5】 前記表示情報を表示させる表示装置と、

前記表示装置の表示画面上に形成され、可視的に表示された前記仮想実行状態を変化させる操作を該表示画面上で操作するインタフェース部と  
を有することを特徴とする請求項 1 4 に記載のアプリケーションプログラム開発支援システム。

【請求項 1 6】 前記インタフェース部は、

前記仮想実行状態における任意のプログラム実行段階で、前記実行環境からシステムコールを発行させる機能と、

前記仮想実行状態における任意の実行環境の要素に対して、前記実行環境資源からのシステムコールを発行させる機能と  
により該仮想実行状態を変化させるものであることを特徴とする請求項 1 5 に記載のアプリケーションプログラム開発支援システム。

【請求項 1 7】 前記インタフェース部は、

前記仮想実行状態における実行環境上で操作される任意のオブジェクトに対して、前記実行環境からのシステムコールを発行させることにより該仮想実行状態を変化させるものであることを特徴とする請求項 1 5 に記載のアプリケーションプログラム開発支援システム。

【請求項 1 8】 前記インタフェース部は、

前記仮想実行状態における任意のプログラム実行段階で、前記実行環境上で割り込み起動される他のプログラムによるイベントを発行させることにより該仮想実行状態を変化させるものであることを特徴とする請求項 1 5 に記載のアプリケーションプログラム開発支援システム。

【請求項 1 9】 前記インタフェース部は、

前記仮想実行状態における任意のプログラム実行段階で、前記実行環境上で割り込み起動される他のプログラムによるイベントを発行させる機能と、当該他のプログラムの実行に要する時間を任意の長さに設定する機能とにより該仮想実行状態を変化させるものであることを特徴とする請求項 1 5 に記載のアプリケーション

ョンプログラム開発支援システム。

【請求項 2 0】 前記アプリケーションプログラムを前記実行環境源上で実行可能な形態で出力する実行オブジェクト生成部を有することを特徴とする請求項 1 4 記載のアプリケーションプログラム開発支援システム。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、リアルタイムOS (Real-Time Operating System) 等で動作するマルチタスク・プログラムのテストやデバッグなどに利用される情報処理装置、不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体、不具合解析方法、及びアプリケーションプログラム開発支援システムに関する。

【0 0 0 2】

【従来の技術】

TRON (The Real-time Operating system Nucleus) に代表されるリアルタイムOSの適用範囲は、マイクロプロセッサ技術の発展により拡大の一途をたどっており、産業用途のみならず、通信機器やオフィス機器などの業務機器分野、家電や携帯電話などの民生用機器などへも急激に拡大している。

【0 0 0 3】

一般に、このようなリアルタイムOS配下で実行されるマルチタスク方式のアプリケーション・プログラムの不具合解析をするために、各プログラムの動きを追跡するのは容易でない。リアルタイムOSでは、特定のタイミングで動作するタスクを次々と切り替えていくことによって、複数のプログラムを動かすマルチタスク方式が採られているため、一つのプログラムを追うだけでは解析できないからである。

【0 0 0 4】

そのため、プログラムの動作を追跡する機能を有するOSが開発され、プログラムの動作状況を解析するためのひとつの手掛りとして、プログラムの実行履歴情報が用いられている。図4 2に示すように、プログラムの実行履歴情報4は、プログラム動作の追跡機能を有するOS 1 0 1によってOS 1 0 1のメモリ上に

出力されたり、更にデバッガ 1 0 2 等の開発ツールによってデバッガ 1 0 2 等が持つメモリ上に出力されたりする。

#### 【 0 0 0 5 】

図 4 4 に、 $\mu$  I T R O N (Micro Industrial TRON) 仕様のプログラム実行履歴情報を例示する。図 4 4 (a) は、システムコール発行履歴のフォーマット例であり、1 レコードで 1 事象を表し、機能のタイプ (type)、発行元タスク (oid)、システムコールの種類 (sysid)、発行先タスク (obj) で構成されている。出力される実行履歴情報の種類としては、このようなシステムコール発行履歴の他に、タスク切り換え履歴、ハンドラ実行履歴などがある。

#### 【 0 0 0 6 】

図 4 4 (b) は、O S 1 0 1 やデバッガ 1 0 2 等によってメモリ上に出力された実行履歴データの例である。(i) は、タスク切り換えが行われたことを表しており、アイドル状態 (タスク = 0) からタスク 1 (タスク = 1) に制御が移行している。(ii) は、システムコールが発行されたことを表しており、タスク 1 (oid=1) から sta\_tsk (sysid=-19) という種類のシステムコールがタスク 2 (obj=2) に対して発行されている。

#### 【 0 0 0 7 】

##### 【発明が解決しようとする課題】

上述したように、プログラムの実行履歴をデータとして保存することができるようになり、プログラムの不具合を解析するための手掛りとして利用されているが、以下のような問題がある。

#### 【 0 0 0 8 】

第 1 に、コントローラのメモリ容量の関係から、履歴情報を採取できる量には制限があり、実際には図 4 3 のように、履歴情報を細切れに保存することになってしまう。特に、ローコストを目指したシステムの場合、このメモリ容量はかなり限定されてしまう。

#### 【 0 0 0 9 】

第 2 に、出力される履歴情報は、図 4 4 に示したように、主に数字からなる文字の羅列であり、しかも大量に出力されがちであることから、この履歴情報を手

掛りにプログラムの動作軌跡を辿って不具合を解析することは、大変困難である。

【 0 0 1 0 】

その結果、プログラムの不具合解析には、依然として多大な時間を要し、開発コストや開発期間などに大きな影響を与えていた。

【 0 0 1 1 】

また、従来、ある実行環境向けに行うプログラミング段階において、その実行環境管理下にある資源の状態変化を把握することや、プログラミング内容の整合性判定、そして実行環境ハードウェアが持つ割り込み機能等を考慮しながらプログラミングすることはできても、その整合性を確認する術がなく、実際にそのプログラムから作成される実行オブジェクトを、実行環境と同じ条件で実行させ、アプリケーションシステムの振る舞いを確認する方法が取られている。

【 0 0 1 2 】

しかしながら、上述した従来の方法では、単純なプログラミングミスも実行オブジェクトを実際に実行させないと、そのミスが判明しないという問題があった。

【 0 0 1 3 】

そこで、本発明はこのような課題を解決するためになされたものであって、対話形式でユーザーから指摘された不具合箇所とプログラムの実行履歴情報から、不具合要因と解決策を特定し、これをユーザーに提示することができる情報処理装置、及び情報処理装置に搭載される不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体を提供することを目的とする。

【 0 0 1 4 】

本発明の第 2 の目的は、前記特定した不具合要因と解決策から、該不具合を解決したプログラムを生成してユーザーに提示することができる情報処理装置、及び情報処理装置に搭載される不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体を提供することを目的とする。

【 0 0 1 5 】

更に、本発明の第 3 の目的は、プログラミング段階において、その実行環境の

動作規則に従いアプリケーションシステムの振る舞いを視覚的に確認し、早期にプログラミングの不整合発見を行い、作業行程の戻りを低減させることにより、効率的なアプリケーションシステム開発を可能とすることのできるアプリケーション開発システムを提供することを目的とする。

【0016】

【課題を解決するための手段】

上記課題を解決するために、本発明は、プログラムの実行履歴情報をもとに、該プログラムの動作状況を時系列にユーザーに表示する表示手段と、表示された動作状況の中でユーザーが不具合の箇所を指定するための入力手段と、前記入力手段によってユーザーから指定された不具合の箇所と該プログラムの動作状況から該不具合要因を解析し、この不具合要因を解決するための解決策を特定する動作解析手段とを有し、前記動作解析手段は、解決策を反映させた前記動作状況を再作成し、前記表示手段は、前記不具合要因と前記解決策と再作成した動作状況とをユーザーに表示することを特徴とする。

【0017】

このとき、前記解決策を反映させる箇所を特定することができない場合には、更に、ユーザーによって該解決策を反映させる箇所を指定させ、前記動作解析手段は、この指定された箇所に該解決策を反映させた前記動作状況を再作成し、前記表示手段は、前記不具合要因と解決策と再作成した動作状況とをユーザーに表示することを特徴とする。

【0018】

更にまた、前記解決策と前記実行履歴情報とをもとに、本来の仕様を満たすプログラムのスケルトンを自動生成することを特徴とする。

【0019】

本発明により、対話形式により、ユーザーから指摘された不具合箇所とプログラムの実行履歴情報から不具合要因と解決策を特定し、これをユーザーに提示することができる。

【0020】

また、前記特定した不具合要因と解決策から、該不具合を解決したプログラム

を生成して同様にユーザーに提示することができる。

【 0 0 2 1 】

更にまた、本発明は、ハードウェア資源上で実行されるアプリケーションプログラムの開発に用いられる開発支援システムにおいて、前記ハードウェア資源、およびソフトウェア部品を含む実行環境での実行時におけるシステム環境と、該実行環境の動作規則を定義する環境定義部と、環境定義に基づいて前記実行環境におけるアプリケーションプログラムの仮想的な実行状態（以下「仮想実行状態」という。）を検証するチェック部と、前記仮想実行状態を可視的に表示するための表示情報を作成する表示情報作成部とを有することを特徴とする。

【 0 0 2 2 】

本発明においては、前記表示情報を表示させる表示装置と、前記表示装置の表示画面上に形成され、可視的に表示された前記仮想実行状態を変化させる操作を該表示画面上で操作するインターフェース部とを設けることが好ましい。

【 0 0 2 3 】

このような本発明によれば、実行環境の動作規則等の環境定義に従い、実際のハードウェア資源上における実行状態を再現することができ、これによりアプリケーションシステムの振る舞いを検証することができるため、従来方法よりも早期のプログラミングの不整合発見が行うことができ、作業行程の戻りも少なくなり効率的なアプリケーション開発が可能となる。

【 0 0 2 4 】

本発明のその他の目的、特徴、効果は、以下で図面を参照して述べる詳細な説明により、一層明らかになるものである。

【 0 0 2 5 】

【発明の実施の形態】

以下、図面に基づいて、本発明の実施形態について説明する。

【 0 0 2 6 】

図 1 は本発明に係る情報処理装置の一実施形態を示す概略構成図であり、図 2 は本発明に係る情報処理装置に搭載される不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体の処理の流れを示した流れ図である。



## 【 0 0 2 7 】

図 1 において、この情報処理装置は、ユーザー 5 からの入力 1 1 を受付け、又ユーザー 5 に情報を提示 1 2 するユーザーインタフェース部 1 と、ユーザーインタフェース部 1 が受付けたユーザー 5 からの入力情報を解析してタスク動作判定部 3 に送る要求解析部 2 と、実行履歴ファイル 4 に格納されているプログラムの実行履歴情報 1 1 とシステムコール対応テーブル 7 とタスク動作判定部 3 からの情報をもとにプログラムの不具合要因とその解決策を特定し、各種解析結果情報をユーザーインタフェース部 1 に送り、さらに解析結果情報から不具合を解決するプログラムを生成してプログラムファイルに出力するタスク動作判定部 3 とから構成される。

## 【 0 0 2 8 】

続いて、図 2 をもとに、全体の処理の流れを説明する。

## 【 0 0 2 9 】

ユーザーインタフェース部 1 は、予めタスク動作判定部 3 がプログラムの動作履歴情報 1 1 をもとに作成した検証データ 1 4 (図 3) をユーザー 5 に表示する(ST01)。ユーザー 5 は表示された検証データ 1 4 を見て、動作的に不具合がないと確認した場合、処理を終了する(ST04)。一方、ユーザー 5 は表示された検証データ 1 4 の中に不具合を発見した場合、その不具合箇所を指摘する(ST05)。ユーザーインタフェース部 1 は、ユーザー 5 が入力した不具合の箇所 1 2 を、要求解析部 2 に転送する(ST06)。

## 【 0 0 3 0 】

次に、要求解析部 2 は、ユーザーインタフェース部 1 から受取ったユーザー 5 が入力した指摘箇所 1 2 と先にタスク動作判定部 3 が作成した検証データ 1 4 から、比較データ 1 3 (図 4) を作成し(ST07)、作成した比較データ 1 3 をタスク動作判定部 3 に転送する(ST08)。

## 【 0 0 3 1 】

次に、タスク動作判定部 3 は、要求解析部 2 から受取った比較データ 1 3 と先に作成しておいた検証データ 1 4 とを比較して、要求タスクの状態を検索し(ST09)、不具合の要因を解析する(ST10)。タスク動作判定部 3 は不具合の要因を

特定したら、不具合を解決する機能を検証データ 1 4 に反映させた上で、検証データ 1 4 をユーザーインタフェース部 1 に転送する (ST11)。続いて、タスク動作判定部 3 は、判定結果データ 1 (図 5) 及び判定結果データ 2 (図 6) を作成してユーザーインタフェース部 1 に転送する (ST12)。

#### 【 0 0 3 2 】

ここまでで不具合が全て解決されていて (ST13)、且つ処理終了をする場合 (ST15) には、ST11で更新した検証データ 1 4 をもとに不具合を解決した正常プログラム・ソース 1 8 を生成し (ST16)、ユーザーインタフェース部 1 に制御を戻す。一方、未解決の不具合要因が残っている場合 (ST14)、不具合を解決するためにユーザーに問い合わせる不具合解決用質問データ 1 7 (図 7) を作成し、ユーザーインタフェース部 1 に転送し (ST14)、ユーザーインタフェース部 1 に制御を戻す。

#### 【 0 0 3 3 】

以上説明したST01からST16までの処理を、プログラムの不具合が解決されて処理を終了するまで繰り返す。

#### 【 0 0 3 4 】

次に、実際に不具合が存在するプログラム例に基づいて、本発明の不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体を搭載した情報処理装置の処理動作について、更に詳細に説明する。

#### 【 0 0 3 5 】

ここでは、 $\mu$  I T R O N 3 . 0 仕様のリアルタイムOS配下で動作するアプリケーション・プログラムを例に説明する。

#### 【 0 0 3 6 】

##### (第 1 実施例)

本実施例では、アラーム機能付きの時計のプログラムを想定する。本プログラムで使用するタスクは以下の 3 本である。

#### 【 0 0 3 7 】

- (1) startup (task\_ID=1,priority=1) [アラーム設定モード]
- (2) タスク A (task\_ID=2,priority=3) [ノーマルモード]

## (3) タスク B (task\_ID=3,priority=2)

本プログラムは、アラームを設定すると（アラーム設定モードを処理すると）、ノーマルモードに移行し、その後処理を終了するものであり、本来は図 8 に示すタイムチャートのような動作を想定して作成されたアプリケーション・プログラムである。従って、例えば処理終了時点を注目してみると、タスク A（ノーマルモード）で処理が終了する、つまりその時点での最終実行タスクはタスク A である、というのがこのプログラムの本来の仕様である。

## 【0038】

ところが、実際にこのプログラムを動作させてみたところ、図 9 に示すような結果が得られたとする。実行結果の処理終了時点を注目してみると、startup タスク（アラーム設定モード）で処理が終了している。すなわち、このプログラムは本来の仕様を満たしていない、このプログラムには不具合が潜在しているということである。

## 【0039】

このような状況において、ユーザー 5 は本発明の情報処理装置及び情報処理装置に搭載された不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体を使用して、以下のように不具合を解決することができる。

## 【0040】

## [処理 1]

ユーザーインタフェース部 1 の処理の流れを図 10 示す。

## 【0041】

ユーザーインタフェース部 1 は、予めタスク動作判定部 3 がプログラムの動作履歴情報 11 をもとに作成した検証データ 14（図 3）をユーザー 5 に表示する（ST21～ST24）。ユーザー 5 への表示例を図 11 に示す。図 11 のようにプログラムの各タスクの動作とタスクの切り換えの様子を時系列に表せば、ユーザー 5 は表示されたプログラムの動作における不具合を指摘しやすい。特に、数字の羅列である実行履歴情報 11 をグラフィックで表現することは有効である。

## 【0042】

ここで、ユーザー 5 は当該プログラムの不具合箇所を指摘する入力を行う（ST

26)。この場合、図 1 1 に示した位置（最終イベントの後のタスク A の位置）にマウスカーソルを指定したとする。この指定位置が、ユーザー 5 が指摘した不具合の箇所となる。本来のプログラムの仕様では、この時点ではタスク A は実行状態でなければならないはずである。

#### 【 0 0 4 3 】

ユーザーインタフェース部 1 は、ユーザー 5 が入力した不具合の箇所データ 1 2 を、要求解析部 2 に転送（ST27）し、要求解析部 2 に制御を移す。このとき転送する不具合箇所データ 1 2 は、図 1 1 においてマウスカーソルを指定した位置、すなわちイベント順を示していた X 軸とタスクを示していた Y 軸との座標位置で一意に認識することができる。ユーザーが指摘した箇所データ 1 2 は、 $X = 7$ 、 $Y = 3$  である。

#### 【 0 0 4 4 】

続いて、図 1 2 に要求解析部 2 の処理の流れを示す。

#### 【 0 0 4 5 】

要求解析部 2 は、ユーザーインタフェース部 1 から受取った不具合箇所データ 1 2 と先にタスク動作判定部 3 が作成した検証データ 1 4 から、要求内容を解析する。まず、X 座標値と検証データ 1 4 のイベントとを比較し、ユーザー 5 の要求タイミングを特定する（ST28）。次に、Y 座標値と検証データ 1 4 のアイテム順から、ユーザーが要求しているタスクを特定する（ST29）。このときの処理イメージを図 1 3 に示す。すなわち、ユーザー 5 の要求タイミングはイベントの最後（第 7 イベントの位置）、要求タスクはタスク A（3 番目のアイテム）である。

#### 【 0 0 4 6 】

要求解析部 2 は、このようにして比較データ 1 3 を作成し（ST30）、作成した比較データ 1 3 をタスク動作判定部 3 に転送して、制御をタスク動作判定部 3 に移す。尚、このときの検証データ 1 4 と比較データ 1 3 を図 1 6 に示す。

#### 【 0 0 4 7 】

##### 〔第 1 次要求〕

この場合の要求タスクはタスク A であり、要求タイミングはイベントの最後で

ある。

【 0 0 4 8 】

〔処理 2〕

続いて、図 1 4 にタスク動作判定部 3 の処理の流れを示す。

【 0 0 4 9 】

タスク動作判定部 3 は、要求解析部 2 から受取った比較データ 1 3 と先に作成しておいた検証データ 1 4 とを比較して、要求タスクの状態を検索する (ST31)。図 1 6 に示す検証データ 1 4 を検索してみると、イベントの最後の時点でのタスク A の状態は、「実行可能 (READY)」状態であることが判る (ST32)。「実行可能」状態にあるタスクとは、実行する準備は整っているが、自分より優先度の高いタスクもしくは優先度が自分と同じであるタスクが「実行 (RUNNING)」状態であるために、実行できない状態にある、言いかえると、実行できる状態のタスクの中で自分が最高の優先順位になればいつでも実行できる状態にあるタスクである。

【 0 0 5 0 】

本発明の情報処理装置の不具合解析処理が終了するのは (目的を達成するのは)、ユーザーが第 1 次要求で指摘したタスク、すなわちタスク A がイベントの最後の時点で「実行」状態になることである。

【 0 0 5 1 】

タスク A を「実行」状態にするまで (不具合要因を特定するまで)、以下に説明する判定結果データ 1、判定結果データ 2 を作成し、作成した判定結果データから本来あるべき処理を特定し、それを検証データ 1 4 に反映させる。このときの各データの状態を図 1 7 に示す (但し、図 1 7 は処理完了状態の図である)。

【 0 0 5 2 】

まず、N の値を判定結果データ 1 の (1) 欄と判定結果データ 2 の (1) 欄に入れる (ST33)。ここでは第 1 次要求であるため、N の値は「1」である。判定結果データ 1 の (1) 欄と判定結果データ 2 の (1) 欄のそれぞれの欄に「1」が入る。

【 0 0 5 3 】

次に、判定結果データ 1 の (2) 欄に要求タスクであるタスク A を入れ、判定結果データ 1 の (3) 欄にタスク A の状態を入れる (ST34)。

## 【 0 0 5 4 】

次に、検証データ 1 4 から、イベントの最後の時点で「実行」であったタスクを検索する (ST35)。この場合、イベントの最後の時点で「実行」であったタスクは、startup タスクである。検索したこのタスクを判定結果データ 2 に入れる (ST36)。

## 【 0 0 5 5 】

次に、判定結果データ 2 に入れたタスク (この場合、startup タスク) が「ext\_tsk」を発行した場合の検証データ 1 4 を作成する (ST37)。具体的には、7 番目のイベントに、startup タスクが「ext\_tsk」を発行するというイベントを検証データ 1 4 に作成するのである。(必然的に、第 8 イベントとしてタスク切り換え (「Task Dispatch」) が発生する。)

そして、このイベント (第 7 ~ 8 イベント) を検証データ 1 4 に反映させたことで、第 1 次要求のタスク A が「実行」状態に遷移したかどうかを、検証データ 1 4 から検索するのである (ST38)。

## 【 0 0 5 6 】

以上の ST33 ~ ST39 までの処理を、第 1 次要求のタスク A が「実行」状態になるまで繰り返す (ST39)。

## 【 0 0 5 7 】

この場合に、「実行」状態に遷移したのは、タスク A ではなく、タスク B である。従って、再び ST33 に戻って、タスク A がイベントの最後の時点で「実行」状態になるための処理を行う。

## 【 0 0 5 8 】

## 〔処理 3〕

まず、N の値を判定結果データ 1 の (1) 欄と判定結果データ 2 の (1) 欄に入れる (ST33)。ここでは第 1 次要求であるため、N の値は「1」である。それぞれの欄に「1」が入る。

## 【 0 0 5 9 】

次に、判定結果データ 1 の (2) 欄に要求タスクであるタスク A を入れ、判定結果データ 1 の (3) 欄にタスク A の状態を入れる (ST34)。検証データ 1 4 から、タスク A の状態は「実行可能」であることが判る。

#### 【 0 0 6 0 】

次に、検証データ 1 4 から、イベントの最後の時点で「実行」状態であったタスクを検索する (ST35)。この場合、イベントの最後の時点で「実行」状態であったタスクはタスク B である。検索したこのタスク B を判定結果データ 2 に入れる (ST36)。

#### 【 0 0 6 1 】

次に、判定結果データ 2 に入れたタスク (この場合、タスク B) が「ext\_tsk」を発行した場合の検証データ 1 4 を作成する (ST37)。具体的には、9 番目のイベントに、タスク B が「ext\_tsk」を発行するというイベントを検証データ 1 4 に作成するのである。(必然的に、第 10 イベントとしてタスク切り換え (「Task Dispatch」) が発生する。)

そして、このイベント (第 9 ~ 10 イベント) を検証データ 1 4 に反映させたことで、第 1 次要求のタスク A が「実行」状態に遷移したかどうかを、検証データ 1 4 から検索するのである (ST38)。

#### 【 0 0 6 2 】

この場合は、タスク A が「実行」の状態に遷移したので、処理終了のフラグ等をセットして (ST51)、イベント 7 ~ 10 を反映させた検証データ 1 4 から今回の不具合を解決したプログラムのスケルトンを生成し、プログラムファイル 6 に出力する (ST52)。このときのプログラムの生成例を図 1 8 に示す。図 1 8 の (a) 及び (b) に示すステップが、今回の不具合要因とその解決策であった。

#### 【 0 0 6 3 】

図 1 7 に示すように、第 1 次要求のタスク A が「実行」状態に至るまで要した判定結果データ 1、判定結果データ 2 と検証データ 1 4 を、ユーザーインタフェース部 1 に転送し、制御をユーザーインタフェース部 1 に戻す。

#### 【 0 0 6 4 】

ユーザーインタフェース部 1 は、タスク動作判定部 3 から転送された検証デー

タ 1 4、判定結果データ 1、判定結果データ 2 をユーザーに表示する。これにより、今回の不具合の要因とその解決策をユーザー 5 に提示することができ、また、本来の仕様に沿ったプログラムの動作をユーザー 5 に提示することができる。

【 0 0 6 5 】

また、本来の仕様を満たしたプログラムがプログラムファイル 6 に出力されているので、ユーザー 5 はこれを利用することができる。

【 0 0 6 6 】

(第 2 実施例)

本実施例で想定するプログラムが使用するタスクは以下の 4 本である。

【 0 0 6 7 】

- (1) startup (task\_ID=1,priority=1)
- (2) タスク A (task\_ID=2,priority=2)
- (3) タスク B (task\_ID=3,priority=2)
- (4) タスク C (task\_ID=4,priority=2)

(使用するセマフォ：Semaphore (ID=1,初期セマフォカウンタ=0) )

本プログラムは、第 1 実施例と同様に、本来は図 1 9 に示すタイムチャートのような動作を想定して作成されたアプリケーション・プログラムである。従って、例えば処理終了時点を注目してみると、タスク B で処理が終了する、つまりその時点での最終実行タスクはタスク B である、というのがこのプログラムの本来の仕様である。

【 0 0 6 8 】

ところが、実際にこのプログラムを動作させてみたところ、図 2 0 に示すような結果が得られたとする。実行結果によると、「Idle Mode」(レディーキューにタスクがない状態)になっている。

【 0 0 6 9 】

このような状況において、ユーザー 5 は本発明の情報処理装置及び情報処理装置に搭載された不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体を使用して、以下のように不具合を解決することができる。

【 0 0 7 0 】



## 〔処理 1〕

図 1 0 のフローチャートに沿って、処理の流れを説明する。

## 【0 0 7 1】

ユーザーインタフェース部 1 は、予めタスク動作判定部 3 がプログラムの動作履歴情報 1 1 をもとに作成した検証データ 1 4 をユーザー 5 に表示する（ST21～ST24）。

## 【0 0 7 2】

ここで、ユーザー 5 は不具合箇所を指摘する入力を行う（ST26）。この場合、実施例 1 と同様に、ユーザー 5 はイベントの最後を指摘したものとする。この指定位置が、ユーザー 5 が指摘した不具合の箇所となる。本来のプログラムの仕様では、この時点ではタスク B は実行状態でなければならないはずである。

## 【0 0 7 3】

ユーザーインタフェース部 1 は、ユーザー 5 が入力した不具合の箇所データ 1 2 を、要求解析部 2 に転送（ST27）し、要求解析部 2 に制御を移す。ユーザーが指摘した箇所データ 1 2 は、 $X = 12$ 、 $Y = 4$  である。

## 【0 0 7 4】

要求解析部 2 の処理の流れを図 1 2 のフローチャートに沿って説明する。

## 【0 0 7 5】

要求解析部 2 は、ユーザーインタフェース部 1 から受取った不具合箇所データ 1 2 と先にタスク動作判定部 3 が作成した検証データ 1 4 から、要求内容を解析する。まず、 $X$ 座標値と検証データ 1 4 のイベントとを比較し、ユーザー 5 の要求タイミングを特定する（ST28）。次に、 $Y$ 座標値と検証データ 1 4 のアイテム順から、ユーザーが要求しているタスクを特定する（ST29）。この場合、ユーザー 5 の要求タイミングはイベントの最後（第 1 2 イベントの位置）、要求タスクはタスク B（4 番目のアイテム）である。

## 【0 0 7 6】

要求解析部 2 は、このようにして比較データ 1 3 を作成し（ST30）、作成した比較データ 1 3 をタスク動作判定部 3 に転送して、制御をタスク動作判定部 3 に移す。尚、このときの検証データ 1 4 と比較データ 1 3 を図 2 1 に示す。

【 0 0 7 7 】

〔第 1 次要求〕

この場合のユーザー 5 の要求タスクはタスク B であり、要求タイミングはイベントの最後である。

【 0 0 7 8 】

〔処理 2〕

続いて、タスク動作判定部 3 の処理の流れを図 1 4 のフローチャートに沿って説明する。

【 0 0 7 9 】

タスク動作判定部 3 は、要求解析部 2 から受取った比較データ 1 3 と先に作成しておいた検証データ 1 4 とを比較して、要求タスクの状態を検索する (ST31)。図 2 1 に示す検証データ 1 4 を検索してみると、要求タイミングである「イベントの最後」の時点でのタスク B の状態は、「待ち (WAITING)」状態であることが判る (ST32、ST40)。「待ち」状態にあるタスクとは、そのタスクを実行できる何らかの条件が整わないために実行できない状態にある、言いかえると、何らかの条件が満たされるのを待っている状態にあるタスクである。

【 0 0 8 0 】

本発明の情報処理装置の不具合解析処理が終了するのは (目的を達成するのは)、ユーザーが第 1 次要求で指摘したタスク、すなわちタスク B がイベントの最後の時点で「実行」状態になることである。

【 0 0 8 1 】

この場合には、タスク B の状態を「実行」状態にする前に、まず「実行可能」状態にする必要がある。「待ち」状態にあるタスクは、直接「実行」状態には移行できないためである。その処理を以下に説明する。

【 0 0 8 2 】

まず、N の値を判定結果データ 1 の (1) 欄と不具合解決用質問データの (1) 欄に入れる (ST45)。ここでは第 1 次要求であるため、N の値は「1」である。それぞれの欄に「1」が入る。

【 0 0 8 3 】

次に、判定結果データ 1 の (2) 欄に要求タスクであるタスク B を入れ、判定結果データ 1 の (3) 欄にタスク B の状態 (「待ち」) を入れる (ST46)。

## 【 0 0 8 4 】

次に、この第 1 次要求の要求タスク B を「実行可能」状態に遷移させることができるシステムコールをシステムコール対応テーブル 7 から検索する (ST47)。このシステムコール対応テーブル 7 には、解決策を特定するためのプログラム機能対応情報が格納されており、その一例を図 1 5 に示す。

## 【 0 0 8 5 】

この場合、図 2 1 の検証データ 1 4 によると、タスク B は「slp\_tsk」によって「(起床) 待ち」の状態にあるため、図 1 5 のシステムコール対応テーブル 7 から「slp\_tsk」に対応するシステムコールを検索するのである。図 1 5 のシステムコール対応テーブル 7 を検索すると、2 番目のエントリに「slp\_tsk」があり、それに対応するシステムコールは「wup\_tsk」である。この「wup\_tsk」を発行すれば、タスク B の状態を「待ち」から「実行可能」に遷移させることができるのである。検索結果の「wup\_tsk」を不具合解決用質問データの (3) 欄に入れる (ST48)。

## 【 0 0 8 6 】

次に、不具合解決用質問データの (2) 欄に、タスク B の状態を「待ち」から「実行可能」に遷移させるシステムコールである「wup\_tsk」の発行対象タスクを入れる (ST49)。この場合、発行対象タスクはタスク A である。

## 【 0 0 8 7 】

この「wup\_tsk」を「何時」「何処」で発行するか、ということは機械的には決められない。発行するタイミング、発行する場所によって、処理が変わってしまうからである。従って、このシステムコール「wup\_tsk」を「何時」「何処」で発行するべきなのかをユーザー 5 に問合せる。この問合せのためのデータが不具合解決用質問データなのである。

## 【 0 0 8 8 】

この不具合解決用質問データによる問合せを第 2 次要求とする (N に 1 加算する (ST50))。ユーザーインタフェース部 1 に制御を戻し、ユーザー 5 に検証デ

ータと不具合解決用質問データを提示し、質問に対する回答の入力を促す。

【0089】

〔第2次要求〕

この場合、ユーザー5が要求した要求タスクはタスクA、要求タイミングは「システムコール「wai\_sem」を発行した後」であったとする。

【0090】

〔処理3〕

ユーザーインタフェース部1に戻って、ユーザー5が入力した箇所のデータ12を、要求解析部2に転送（ST27）し、要求解析部2に制御を移す。このとき転送する箇所データ12は、 $X=6$ ， $Y=3$ である。

【0091】

要求解析部2は、比較データ13を作成し（ST30）、作成した比較データ13をタスク動作判定部3に転送して、制御をタスク動作判定部3に移す。尚、このときの検証データ14と比較データ13を図22に示す。

【0092】

タスク動作判定部3は、要求解析部2から受取った比較データ13と検証データ14とを比較して、要求タスクの状態を検索する（ST31）。図22に示す検証データ14を検索してみると、要求タイミングでのタスクAの状態は、「（セマフォ資源の獲得）待ち」状態であることが判る（ST32、ST40）。この場合には、タスクAの状態を「待ち」から「実行可能」にする必要がある。その処理を以下に説明する。

【0093】

まず、Nの値を判定結果データ1の（1）欄と不具合解決用質問データの（1）欄に入れる（ST45）。ここでは第2次要求であるため、Nの値は「2」である。それぞれの欄に「2」が入る。

【0094】

次に、判定結果データ1の（2）欄に要求タスクであるタスクAを入れ、（3）欄にタスクAの状態（「待ち」）を入れる（ST46）。

【0095】

次に、この第 2 次要求の要求タスク A を「実行可能」状態に遷移させることができるシステムコールをシステムコール対応テーブル 7 から検索する (ST47)。対応するシステムコールは「sig\_sem」である。この「sig\_sem」を発行すれば、タスク B の状態を「待ち」から「実行可能」に遷移させることができるのである。検索結果の「sig\_sem」を不具合解決用質問データの (3) 欄に入れる (ST48)。

## 【 0 0 9 6 】

次に、不具合解決用質問データの (2) 欄に、タスク B の状態を「待ち」から「実行可能」に遷移させるシステムコールである「sig\_sem」の発行対象タスクを入れる (ST49)。この場合、発行対象タスクはタスク C である。

## 【 0 0 9 7 】

この「wup\_tsk」を「何時」「何処」で発行するか、ということも機械的には決められない。発行するタイミング、発行する場所によって、処理が変わってしまうからである。従って、このシステムコール「sig\_sem」を「何時」「何処」で発行するべきなのかをユーザー 5 に問合せる。

## 【 0 0 9 8 】

この不具合解決用質問データによる問合せを第 3 次要求とする。N に 1 加算して、ユーザーインタフェース部 1 に制御を戻し (ST50)、ユーザー 5 に検証データと不具合解決用質問データを提示し、質問に対する回答の入力を促す。

## 【 0 0 9 9 】

## 〔第 3 次要求〕

この場合、ユーザー 5 が要求した要求タスクはタスク C、要求タイミングは「システムコール「ext\_tsk」を発行する前」であったとする。

## 【 0 1 0 0 】

## 〔処理 4〕

ユーザーインタフェース部 1 に戻って、ユーザー 5 が入力した箇所のデータ 1 2 を、要求解析部 2 に転送 (ST27) し、要求解析部 2 に制御を移す。

## 【 0 1 0 1 】

要求解析部 2 は、比較データ 1 3 を作成し (ST30)、作成した比較データ 1 3

をタスク動作判定部 3 に転送して、制御をタスク動作判定部 3 に移す。尚、このときの検証データ 1 4 と比較データ 1 3 を図 2 3 に示す。

【 0 1 0 2 】

タスク動作判定部 3 は、要求解析部 2 から受取った比較データ 1 3 と検証データ 1 4 とを比較して、要求タスクの状態を検索する (ST31)。図 2 2 に示す検証データ 1 4 を検索してみると、要求タイミング「システムコール「ext\_tsk」を発行する前」時点でのタスク C の状態は、「実行」状態であることが判る (ST32、ST40)。

【 0 1 0 3 】

このように要求タスクの要求タイミング時点での状態が「実行」状態である場合には、ひとつ前の (N - 1 の) 不具合解決用質問データのシステムコールを要求タイミング時に発行した場合の検証データ 1 4 を作成する (ST41)。本実施例でいうと、不具合解決用質問データの N = 2 の時のシステムコールである「sig\_sem」をタスク C が発行した場合の検証データを作成することになる (図 2 4 の検証データを参照)。

【 0 1 0 4 】

次に、第 2 次要求タスクのイベントの最後の時点での状態を検証データ 1 4 から検索する (ST42、ST43)。

【 0 1 0 5 】

検索の結果、第 2 次要求タスク A は「実行」状態に遷移している。判定結果データ 1 のタスク A の状態を「実行」状態に書きかえる (ST44)。

【 0 1 0 6 】

これで、第 2 次要求タスクの状態が「実行」状態に遷移したことで、第 3 次要求は満たされた。しかし、第 1 要求のタスク B は依然として「実行可能」状態のままであるため、ここまで作成した検証データ、判定結果データ、不具合解決用質問データ (第 1 次分) をユーザーインタフェース部 1 に転送し、制御をユーザーインタフェース部 1 に戻す (ST38~39)。

【 0 1 0 7 】

そして、これまで説明した処理に準じて、第 1 次要求のタスク B が第 1 次要求

タイミングである「イベントの最後」の時点で「実行」状態に遷移するまで、処理を繰り返す。このタスク B が「イベントの最後」の時点で「実行」状態に遷移するまでの検証データ、比較データ、判定結果データ 1、判定結果データ 2、及び不具合解決用データの状態を、図 2 1 ～ 図 2 6 に示す。

## 【 0 1 0 8 】

図 2 6 に示した検証データが本来のプログラムの仕様を満たすものであり、同じく図 2 6 に示した判定結果データ 1，2 と不具合解決質問データが今回の不具合の要因となっていたものである。本発明の情報処理装置は、これらの各データをユーザー 5 に提示することで、プログラムの不具合要因とその解決策を対話形式でユーザーに提示できる。

## 【 0 1 0 9 】

また、処理終了時には、第 1 実施例と同様にして、図 2 6 の検証データからプログラムのスケルトンを自動生成し、ユーザー 5 に提供することができる。

## 【 0 1 1 0 】

以上、第 1 実施例及び第 2 実施例を用いて説明したように、本発明の不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体を搭載した情報処理装置を用いることにより、対話形式で不具合要因とその解決策を容易に特定でき、更に、本来の仕様に沿ったプログラムの動作、及び本来の仕様に沿ったプログラムをも容易に取得することができる。

## 【 0 1 1 1 】

つまり、対話形式でユーザーから指摘された不具合箇所とプログラムの実行履歴情報から不具合要因とその解決策を特定し、更にこの解決策を反映させたプログラムの動作をユーザーに提示することができる。

## 【 0 1 1 2 】

また、前記特定した不具合要因とその解決策から、上記不具合を解決したプログラムを自動生成してユーザーに提示することができる。

## 【 0 1 1 3 】

その結果、プログラムの不具合解析作業を容易にし、従来不具合解析に要していた時間を短縮することができ、プログラムの開発コストや開発期間などを低減

することができる。

【0114】

(第3実施例)

本発明に係るアプリケーションプログラム開発支援システムの実施形態について説明する。なお、本実施形態では、あるマイコン向けの組込み用リアルタイムOS(μITRON3.0仕様)に対応したアプリケーションプログラムを開発する場合を例に説明する。

【0115】

[ハードウェア資源の構成]

図27は、本実施形態に係るアプリケーション開発システム20を実施するためのハードウェア資源の構成を示す概観図である。

【0116】

図27に示すように、ハードウェア資源は、中央処理装置(CPU)51と、RAM52とを備えている。なお、本実施形態では、中央処理装置(CPU)51、RAM52は、バス58を介して、ROM53、通信装置54、補助記憶装置55、表示装置56、入力装置57及び出力装置59に接続されている。

【0117】

ROM53や補助記憶装置55には、コンピュータ・プログラムのコードが記録されている。このコンピュータ・プログラムは、アプリケーション開発システム20上のオペレーティングシステムと協働して中央処理装置51等に命令を与えるものである。コンピュータ・プログラムコードは、RAM52にロードされることによって実行される。なお、このコンピュータ・プログラムのコードは、圧縮され、または、圧縮したコードも含め、複数に分割して、複数の媒体にまたがって記録することもできる。

【0118】

また、本実施形態において、アプリケーション開発システム20へのデータの入出力をするユーザ・インターフェース・ハードウェアとしては、画面位置情報、文字情報を入力するための入力装置57や、画像データをユーザに提示するための表示装置56、出力装置59がある。入力装置57としては、例えば、ポイ



ンティングデバイスやキーボード等がある。この他、データの入出力の方法としては、通信装置 5 4 を介して他のコンピュータ等から入力データを受け取ることや、また補助記憶装置 5 5 などの媒体から行うものが挙げられる。また、表示装置 5 6 は、モニターやプリンターなど、データを可視化して出力するものである。

#### 【 0 1 1 9 】

このように、アプリケーション開発システム 2 0 は、パーソナルコンピュータやワークステーション、携帯情報端末、ネットワークコンピュータ上における実施、あるいはこれらの組合せによるハードウェア構成での実施も可能である。ただし、これらのハードウェア構成は例示であり、全ての構成要素がアプリケーション開発システム 2 0 の必須の構成要素となるものではない。

#### 【 0 1 2 0 】

##### 〔アプリケーション開発システムの構成〕

上述したハードウェア構成によりアプリケーション開発プログラムを実行することによって、かかるハードウェア構成上に本発明のアプリケーション開発システムが仮想的に構築される。図 2 8 は、本実施形態において構築されたアプリケーション開発システムの構成を模式的に示すブロック図である。

#### 【 0 1 2 1 】

同図に示すように、本実施形態に係るアプリケーション開発システムは、環境定義部 3 1 と、プログラミング部 3 2 と、テンプレートファイル生成部 4 0 と、環境定義ファイル生成部 4 3 と、チェック部 4 5 と、実行オブジェクト部 4 6 と、表示情報作成部 4 8 とを有している。

#### 【 0 1 2 2 】

環境定義部 3 1 は、入力された環境定義データを解析し、解析した結果をプログラミング部 3 2 に供給するものである。この環境定義データには、開発するアプリケーションシステムが利用する R T O S 管理下のオブジェクト数や、その初期化設定、開発アプリケーションシステムが稼働するマイコンに発生する外部／内部割り込み要因に対する初期化データ等が含まれる。また、この環境定義部 3 1 は、解析結果から環境定義情報 4 9 を生成する。環境定義情報は、環境定義フ

ファイルを生成する際には、環境定義ファイル生成部 4 3 に入力される。

【 0 1 2 3 】

プログラミング部 3 2 は、入力装置 7 を介して行われるプログラミング操作により開発プログラムを構築するものである。

【 0 1 2 4 】

テンプレートファイル生成部 4 0 は、プログラミング部 3 2 からテンプレートファイル生成の命令を受け取り、プログラミング情報 3 8 の情報を基に、テンプレートファイル 4 2 を生成し、出力するものである。

【 0 1 2 5 】

また、環境定義ファイル生成部 4 3 は、入力された環境定義データをファイル形式に形成するものである。

【 0 1 2 6 】

チェック部 4 5 は、環境定義データに基づいて開発対象であるアプリケーションプログラムを実行させる実行環境の動作規則に従い、仮想的に構築し、この仮想環境の下で開発プログラムの動作チェックを行うものである。具体的には、本実施形態では、システムコール選択チェック部 5 0 と、システムコール発行位置チェック部 3 5 と、システムコール操作対象位置チェック部 3 6 と、R T O S カーネルチェック部 3 7 とを有している。

【 0 1 2 7 】

システムコール選択チェック部 5 0 は、選択されたシステムコールが、プログラミングの対象となる動作環境下において、現実発行しうるか 否かについて判断するものである。

【 0 1 2 8 】

システムコール発行位置チェック部 3 5 は、システムコールを発行すると仮定した位置、もしくはタイミングが、システムコールを現実発行するものか否かについて判断するものである。

【 0 1 2 9 】

システムコール操作対象位置チェック部 3 6 は、システムコールを発行すると仮定した対象について、システムコールが現実発行した場合に、操作できるの

か否かについて判断するものである。

【 0 1 3 0 】

R T O S カーネルチェック部 3 7 は、本実施形態に係るハードウェア資源上で動作する O S である R T O S の動作について、確認するものである。

【 0 1 3 1 】

また、前記実行オブジェクト部 4 6 は、開発されたプログラムをコンパイルし、実行オブジェクトファイル（電子ファイル）として出力するものである。前記表示情報作成部 4 8 は、モニター等の表示装置 6 等において出力可能に、各情報を生成するものである。

【 0 1 3 2 】

[ G U I ( Graphical User Interface ) の構成 ]

次いで、プログラミング装置 3 2 のインタフェースである G U I ( Graphical User Interface ) の構成について説明する。図 2 9 及び図 3 0 は、表示装置 6 に表示された画面構成を示すものである。

【 0 1 3 3 】

この G U I は、表示装置 6 に、ウインドウ 7 0 として表示され、このウインドウ 7 0 は、プログラム中において発行したいシステムコールを選択するシステムコール選択ペイン 6 1 と、プログラム中において利用可能な実行環境提供の資源オブジェクト、および R T O S 管理下のオブジェクト（以下「オブジェクト」という。）／割り込み要因別にグループ別に表示するオブジェクト表示ペイン 6 2 と、プログラミングを行うプログラミング作業ペイン 6 3 とを有している。

【 0 1 3 4 】

プログラミング作業ペイン 6 3 には、システムコール発行位置（オブジェクト）、操作位置（オブジェクト）を指定するために各オブジェクト毎に設けられたシーケンスライン 6 4 と、プログラム中における、理論的なシステムコール発行タイミングや、発行順を定める目盛りペイン 6 5 がインタフェースとして備えられている。

【 0 1 3 5 】

なお、プログラミング作業ペイン 6 3 は、タスク部のプログラミングを行うタ

スク部プログラミング作業ペインと、タスク独立部別に、タスク独立部のプログラミングを行うハンドラ部プログラミング作業ペインから構成される。

#### 【0136】

そして、プログラミング装置32が起動された際に取得された環境定義情報49は、図30に示すように、オブジェクト表示ペイン62に表示される。

#### 【0137】

オブジェクト表示ペイン62は、プログラム中において利用可能なオブジェクト／割り込み要因を表示するタスクオブジェクトグループ82を有している。このタスクオブジェクトグループ82では、周期起動ハンドラグループ85などのグループ分けがなされており、各グループに属する個々のオブジェクトにはオブジェクト名83と、初期状態設定が行われている場合は、オブジェクト初期情報84が併せて表示される。なお、未定義オブジェクトに関しては、オブジェクトグループ名だけが表示される。

#### 【0138】

すなわち、例えば、図中タスクオブジェクト\_\_usr3cdのオブジェクト初期情報は、“ID”がタスクID情報を意味し、“Pri”がタスク優先度情報を意味し、“lst”が初期起動タスク情報（アプリケーションプログラム処理において、最初、実行状態タスクであることを意味する。）を意味する。つまり、本実施形態では、プログラミング作業ペイン63において、プログラミングを始める場合は、\_\_usr3cdが最初にシステムコール発行を行う。

#### 【0139】

また、本実施形態において、システムコール発行操作は、システムコール選択ペイン61から、発行するシステムコールを選択した後に、目盛りペイン65にある目盛り1つに、1つのシステムコールを、その目盛り位置における実行状態タスクのシーケンスライン64上に置くことにより行う。

#### 【0140】

これによりシステムコール発行位置を仮定することができる。プログラミングは、目盛り位置：0を起点として構築し、プログラム処理は目盛り位置：0から順に行われる。また、本実施形態において、この目盛りペイン65には、目盛り

間を開けてシステムコールを発行することを防ぐ機能が装備されている。

#### 【0 1 4 1】

発行したシステムコールが、操作対象オブジェクト指定を必要とする場合は、操作対象オブジェクトが有するシステムコール発行位置と同一目盛り位置のシーケンスライン 6 4 上を指定する。この位置が、システムコール操作対象位置と仮定される。システムコール操作対象位置選択時、システムコール発行位置とは違う目盛り位置を選択できない機能を装備している。

#### 【0 1 4 2】

システムコール発行の操作には、発行時のプログラミング作業ペイン 6 3 において、発行可能目盛りの最大値に発行する追加操作、プログラミング作業ペイン 6 3 にある発行済みシステムコールの位置に対するシステムコール挿入発行操作、発行済みシステムコールに対する削除操作、発行目盛り位置移動操作（複数のシステムコール発行がある場合）、システムコール種別変更操作、システムコール引数変更操作、システムコール操作対象位置移動等の編集機能が設けられている。

#### 【0 1 4 3】

##### 〔アプリケーション開発システムの処理手順〕

そして、上述したアプリケーション開発システム 2 0 は、以下のように動作する。図 3 1 は、本実施形態に係るアプリケーション開発システム 2 0 の動作を示すフロー図である。

#### 【0 1 4 4】

まず、最初に必要なデータとして、入力装置 5 7 から環境定義装置 3 1 に対し、環境定義データを入力する（S 1 0 1）。

#### 【0 1 4 5】

環境定義装置 3 1 に必要なデータが揃うと、入力装置 5 7 から環境定義装置 3 1 に対し、プログラミング装置 3 2 を起動する情報を入力する。入力を受け付けた環境定義装置 3 1 は、プログラミング装置 3 2 を起動する（S 1 0 2）。この起動の際、プログラミング装置 3 2 のインタフェースである G U I（Graphical User Interface）が、表示装置 5 6 に表示される。

【 0 1 4 6 】

プログラミング装置 3 2 は、起動された後、入力装置 5 7 からプログラミング操作入力を待つ。ユーザーは、入力装置 5 7 を介してプログラミング操作によりプログラミングデータの入力を行う（S 1 0 3）。

【 0 1 4 7 】

次いで、入力されたプログラムについて、チェック部 4 5 によりプログラミング情報のチェックを行う（S 1 0 4）。

【 0 1 4 8 】

そして、チェックの結果がエラーであるか否かの判定を行い（S 1 0 5）、エラーである場合にはエラー情報 4 1 を更新するとともに、そのエラーの内容について表示情報作成部 4 8 により表示情報を作成し、表示装置 5 6 において表示する。ステップ S 1 0 5 において、エラーが発生していないと判断された場合には、プログラミング情報 3 8 及びオブジェクト情報 3 9 を保存する（S 1 0 6 及び S 1 0 7）。

【 0 1 4 9 】

その後、プログラミングを終了するか否かの選択を待ち（S 1 0 8）、プログラミングを続行する場合には、ステップ S 1 0 3 に戻り、再度プログラミングデータの入力を受け付ける。ステップ S 1 0 8 において、プログラミングの終了を選択した場合には、テンプレートファイル生成部 4 0 においてテンプレートファイル 4 2 を生成し、出力するとともに（S 1 1 0）、実行オブジェクト生成部 4 6 において、実行オブジェクトファイル 4 7 を生成し、出力し（S 1 1 1）、作業を終了する（S 1 1 2）。

【 0 1 5 0 】

プログラミングデータの入力を受け付けてから、入力結果を表示するまでの処理手順を図 3 2 に示す。この処理手順に沿って、表示装置 5 6 に表示されているウィンドウ 7 0 をインタフェースとするプログラミングを行う。

【 0 1 5 1 】

次いで、発行されるシステムコールを、システムコール選択ペイン 6 1 で入力装置 7 を通じて選択する（S 2 0 1）。また、入力装置 5 7 を通じて、システム

コールが発行する位置を入力する（S 2 0 2）。

【 0 1 5 2 】

入力されたこれらのデータに基づいてシステムコール発行位置チェック部において、システムコール発行位置をチェックする（S 2 0 3）。

【 0 1 5 3 】

次いで、このチェック結果について判定を行い、エラーが発生していない場合には、発行システムコールの操作対象の選択が必要か否かの選択を要求する（S 2 0 5）。捜査対象の選択が不要である場合には、R T O Sカーネルチェック装置にて、発行システムコールカーネル部を処理する（S 2 0 1）。

【 0 1 5 4 】

ステップ S 2 0 5 において発行システムコールの操作対象の選択が必要である場合には、入力装置 5 7 を介して、プログラミング操作システムコール操作外傷位置を入力する（S 2 0 7）。

【 0 1 5 5 】

システムコール操作対象位置チェック部 3 6 にて、システムコール操作対象位置をチェックする（S 2 0 7）。このチェック結果の内容についてエラーが生じているか否かについて判定を行い（S 2 0 8）、エラーが生じていない場合には、入力装置 7 によりプログラミング操作のシステムコール引数を入力し（S 2 0 9）、R T O Sカーネルチェック部 3 7 にて、発行システムコールカーネル部を処理する（S 2 1 0）。

【 0 1 5 6 】

ステップ S 2 1 0 において発行システムコールカーネル部を処理した後、カーネル処理の内容を判定する（S 2 1 1）。このステップ S 2 1 1 においてエラーが発生していない場合には、プログラミング情報 3 8 の保存（S 2 1 2）及びオブジェクト情報 3 9 の保存（S 2 1 3）を行い、表示情報作成部 4 8 にて表示情報を作成し（S 2 1 4）、作成結果を表示装置 5 6 において表示し（S 2 1 5）、終了する。

【 0 1 5 7 】

なお、ステップ S 2 0 4 や S 2 0 8、S 2 1 1 においてエラーが生じた場合に

は、エラー情報 4 1 を保存した後（S 2 1 6）、このエラー情報を表示し（S 2 1 5）、処理を終了する。

【0 1 5 8】

〔システムコール発行時の動作〕

次いで、システムコール時の動作について説明する。ここでは、プログラミング作業ペイン 6 3 がタスク部になっていると仮定し、実行状態タスクである”\_\_usr3cd”がシステムコール“sta\_\_task”を発行し、休止状態タスクである”\_\_usr6cd”を起動する追加操作を例に説明する。図 3 3 は、システムコール時における表示装置 5 6 に表示された画面構成を示す説明図である。

【0 1 5 9】

まず、前述したシステムコール選択ペイン 6 1 から、発行するシステムコールを選択する。この選択操作時、システムコール選択チェック部 5 0 が、プログラミング作業ペイン 6 3 をタスク部に仮定している場合は、タスク部から発行可能なシステムコールを選択可能にし、ハンドラ部に仮定している場合は、ハンドラ種別により、そのハンドラから発行可能なシステムコールが選択可能となるように、ユーザのシステムコール選択操作を監視している。

【0 1 6 0】

本実施形態では、プログラミング作業ペイン 6 3 がタスク部と仮定しているため、システムコールペイン 6 1 からハンドラ部からのみ発行可能なシステムコールを選択することはできない。

【0 1 6 1】

入力装置 7 から、システムコール選択ペイン 6 1 より“sta\_\_task”を選択する。選択後、プログラミング作業ペイン 6 3 において、入力装置 5 7 からシステムコール発行操作を行う。この時点において、システムコールを発行するメモリ位置は、発行済みシステムコールが存在しないため「0」であり、メモリ位置：0 における実行状態タスクは”\_\_usr3cd”であるため、”\_\_usr3cd”タスクのシーケンスライン 6 4 上付近（シーケンスライン上付近とは、対象シーケンスラインの上、あるいは下のシーケンスラインから対象シーケンスラインに近い位置までとする）システムコール発行位置 9 1 を入力装置 7 から選択



する。

#### 【0162】

この追加操作におけるシステムコール発行位置の選択は、先の説明の通り、目盛り間を開けてシステムコールを発行することを防ぐ機能が働く。機能は、図30のようにシステムコール発行選択位置91が指す場所を指定しなくても、目盛り位置：0以降、“\_usr3cd”のシーケンスライン64上付近を選択すれば、目盛り位置：0を選択したことになる。

#### 【0163】

選択（入力）を受け付けたプログラミング装置32は、システムコール発行位置チェック装置35に、選択内容をチェックさせる。システムコール発行位置チェック装置35では、図34の処理手順によりチェックを行う。

#### 【0164】

すなわち、図34に示すように、先ずシステムコール発行位置におけるオブジェクトの種別を判定し（S301）、“OK”であれば、システムコール発行位置におけるオブジェクトの状態について判断する（S302）。このステップS302において“OK”であるときは、戻り値「ret」に、システムコールが操作対象オブジェクトを必要とするか、しないかについての情報を与えて（S303）、処理を終了する。

#### 【0165】

この戻り値「ret」は、システムコールが操作対象オブジェクトを必要とするか、いないかの情報と、エラー情報をセットするものである。本実施形態では、この戻り値「ret」は、32bit領域を持つ変数であり、最上位ビットにはシステムコールが操作対象オブジェクトを必要とするか、しないかの情報がセットされ、それ以外のビットにはエラー情報がセットされる領域に用いられている。

#### 【0166】

ステップS301またはステップS302において“NG”と判断されたときには、戻り値「ret」に、エラー情報を与え、前記ステップS303を経て、処理を終了する。

#### 【0167】

そして、このようにして得られた戻り値「ret」により、発行したシステムコールが“sta\_\_tsk”であり、プログラミング装置 3 2 は、システムコール発行位置チェック装置 3 5 が設定した、ret 情報領域のチェック結果から、チェック判定にエラーが無いことと、“sta\_\_tsk”に操作対象オブジェクトを必要とすることが判明する。その操作対象タスクは“usr6cd”であるため、“usr6cd”タスクのシーケンスライン 6 4 上付近システムコール発行位置 9 1 ‘を入力装置 2 7 から選択する。

## 【0168】

もし、システムコール発行位置 9 1 が図 3 3 と違う場所、例えば、周期起動ハンドラオブジェクトである“cyc\_Hdr1”のシーケンスライン 6 4 付近をシステムコール発行位置として選択した場合には、システムコール発行位置チェック装置 3 5 では、RTOS 仕様から“sta\_\_tsk”発行位置オブジェクトとして不適と判断し、ret 情報領域にエラー情報を与える。

## 【0169】

また、システムコール発行位置 9 1 以外のタスクオブジェクトのシーケンスライン 6 4 付近をシステムコール発行位置として選択した場合は、システムコール発行位置チェック装置 3 5 では、RTOS 仕様からシステム発行タスクの状態不適（実行状態タスクではない）と判断し、ret 情報領域にエラー情報を与える。

## 【0170】

システムコール操作対象位置選択は、システムコール発行位置からマウス等の入力装置によるドラッグ操作で位置を選択する。この際に、先に述べた様に、システムコール操作対象位置は、システムコール発行位置と同一目盛り位置のシーケンスライン上付近を選択させる機能があり、その機能は、図 3 3 中の表示ライン 9 3 のように、システムコール発行位置からのドラッグ操作に、カーソルの縦方向移動にのみ線が追従描画され、その線の末端が選択位置となる様になっている。

## 【0171】

選択（入力）を受け付けたプログラミング装置 3 2 は、システムコール操作対

象位置チェック装置 3 6 に、選択内容をチェックさせる。システムコール操作対象位置チェック装置 3 6 では、図 3 5 の処理手順によりチェックを行う。

#### 【0 1 7 2】

すなわち、図 3 5 に示すように、まず、システムコール操作対象位置のシステムコール操作対象位置におけるオブジェクトの種別を判定し（S 4 0 1）、ここにおいて”OK”であれば、システムコール操作対象位置におけるオブジェクトの状態について判断する（S 4 0 2）。このステップ S 4 0 2 において”OK”であるときは、戻り値「ret」の最上位ビット領域をクリアしてその値を「0」とし（S 4 0 3）、処理を終了する。

#### 【0 1 7 3】

ステップ S 3 0 1 またはステップ S 3 0 2 において”NG”と判断されたときには、戻り値「ret」に、エラー情報を与え、前記ステップ S 3 0 3 を経て、処理を終了する。

#### 【0 1 7 4】

これらの処理により、この戻り値「ret」の最上位ビットにセットされたシステムコールが操作対象オブジェクトを必要とするかしないかの情報がクリアされ、それ以外のビットにはエラー情報がセットされることとなる。

#### 【0 1 7 5】

チェックにおいても、システムコール発行位置のチェックと同じく、システムコール操作対象位置 9 1' が図 3 3 と違う場所、例えば、周期起動ハンドラオブジェクトである”cyc\_Hdr1”のシーケンスライン 6 4 付近をシステムコール操作対象位置として選択した場合は、システムコール操作対象位置チェック装置 3 6 では、RTOS 仕様から”sta\_task”操作対象オブジェクトとして不適と判断し、ret 情報領域にエラー情報を与える。

#### 【0 1 7 6】

また、システムコール操作対象オブジェクトが 9 1' 以外のタスクオブジェクトのシーケンスライン 6 4 付近をシステムコール操作対象位置として選択したが、選択したタスクが休止状態タスクでない場合は、システムコール操作対象位置チェック装置 3 6 では、RTOS 仕様からシステムコール操作対象タスクの状態

不適と判断し、r e t 情報領域にエラー情報を与える。

【0 1 7 7】

システムコール操作対象選択位置のチェックが終了し、エラーが無い場合は、発行したシステムコールが持つ機能の処理をR T O Sカーネルチェック装置3 7にて行う。本実施例においては、休止状態である” \_ u s r 6 c d ” を、実行可能状態にする。この処理におけるエラー判定基準は、R T O S仕様に基づく。

【0 1 7 8】

システムコール発行の操作、そのチェック完了後、プログラミング装置3 2は、ウインドウ7 0に発行結果を追加描画するために、プログラミング情報3 8、オブジェクト情報3 9を更新し、表示情報作成装置4 8にて表示情報を作成し、表示装置6に、図3 6の様な描画更新されたウインドウ7 0が表示される。

【0 1 7 9】

プログラミング情報3 8には、目盛り位置：0に発行されたシステムコールと、その発行内容が情報として保存される。

【0 1 8 0】

オブジェクト情報3 9には、目盛り位置：0に発行されたシステムコールの影響により、オブジェクトがどの様に変化したか、変化したオブジェクト情報だけを更新する。

【0 1 8 1】

オブジェクト情報3 9を利用し、プログラミング装置3 2は、オブジェクト情報内容3 9に変更がある／ないに関わらず、更新処理終了後、オブジェクト情報3 9から、次の目盛り位置における実行状態タスク情報を検出する。その情報を実行状態ナビゲートライン1 0 0として図3 6の様に、発行したシステムコールのシステムコール発行位置から次の目盛り位置まで線描画する。

【0 1 8 2】

また、入力装置5 7のマウス等の操作ポイントを、プログラミング作業ペイン6 3内にある、発行済みシステムコールの描画を外し、あるオブジェクトが持つシーケンスライン6 4上付近に移動させると、操作ポイントが指したプログラム処理の流れの中において、そのオブジェクトの状態をウインドウ7 0に表示させ

、参照することができる。

【0183】

プログラミング装置32起動後、入力装置57から入力した操作によって“s t a \_ t s k”発行のプログラミングが完了する。この装置を利用することにより、システムコール発行の内容を開発装置に実装されているチェック装置群45が、利用するRTOS仕様に従ったチェックと、RTOS管理下のオブジェクト操作を行うため、プログラミングの段階において、システムの不具合除去、不具合検知を行うことが可能となり、オブジェクト情報39の利用により、プログラミング段階において、タスクオブジェクトの状態遷移の表示、他アプリケーションシステムにおいて使用するオブジェクトの状態を表示することにより、プログラミングを強力にサポートすることができる。

【0184】

〔イベント発生時の動作〕

次いで、イベント発生時の動作について説明する。ここでは、アプリケーション開発システム20において、図37の様なプログラミングを入力装置57から入力が行われた場合を例に説明する。図37は、本発明装置によるプログラミング中の状態である。

【0185】

本実施例では、開発アプリケーションシステムが稼働するハードウェアにおいて、ハードウェアの周期的な割り込み、非同期な割り込みイベントの発生時、その割り込みイベントに連動し処理される単位をプログラミングした場合に、プログラミングの段階において、その割り込みイベント発生のタイミングを定義し、アプリケーションプログラムが使用するタスクの振る舞い、およびオブジェクトの状態変化を検証する方法を示す。

【0186】

図37に示すように、タスク部プログラミング作用ペイン111では、タスク部のプログラミングが行われており、c y c \_ H d r l ハンドラ部プログラミング作業ペイン112では、タスク独立部であるc y c \_ H d r l ハンドラ内にプログラミングを、それぞれ別の作業ペインで行っている。

## 【 0 1 8 7 】

タスク独立部は、ハードウェアの周期的な割り込み、非同期な割り込みイベント発生時の処理単位であり、タスク部のプログラミングと同期が取られていない。

## 【 0 1 8 8 】

本実施形態では、図 3 8 の様に、アプリケーションプログラミング構築段階において、タスク独立部である周期起動ハンドラ部の処理単位をタスク部のプログラム中に仮想的に処理させる、割り込み発生イベントを発行することにより、ハードウェアの周期的な割り込み、非同期な割り込みイベントが発生した場合のシミュレーションを検証することができる。

## 【 0 1 8 9 】

割り込み発生イベントを発行し、ハードウェアの周期的な割り込み、非同期な割り込みイベントの発生シミュレーションを検証するためには、まず、タスク部のプログラミング作業ペインにおいて、割り込みイベントを発生させたい目盛り位置に、割り込み発生イベントを発行する。入力装置 5 7 により割り込み発生イベント発行が行われ、入力を受け付けたプログラミング装置 3 2 は、システムコール発行時に行うチェックを同様に行う。チェック内容は、割り込み発生イベント発行位置のオブジェクト種別、オブジェクトの状態である。

## 【 0 1 9 0 】

割り込みイベントを発生させるオブジェクトの状態は、環境定義装置 3 1 に対し、静的に定義するそのオブジェクトに必要なシステム環境定義と、そのオブジェクトの機能を有効／無効にする操作があり、その様な操作が必要な場合のみ入力装置 5 7 から、プログラミング装置 3 2 に対し、そのオブジェクトの動的な操作「システムコール発行」を行う。

## 【 0 1 9 1 】

プログラミング装置 3 2 が、割り込み発生イベントの発行が正しいものであると判断した場合、図 3 8 の処理手順に従い、割り込み発生イベントを発行したハンドラ内のプログラミング処理をすべて行い、その処理の結果、オブジェクトの変化をオブジェクト情報 3 9 に保存し、タスク部の処理に戻ることになる。アプ

リケーションプログラムの処理イメージを、図 4 0 に示す。

#### 【0 1 9 2】

すなわち、ハンドラ内に行われたプログラムを処理するには、図 3 9 に示すように、先ずプログラミング情報 3 8 に保存されている、ハンドラ内の目盛り位置：0 のプログラミング情報から、一つ一つ順番に読み込む（S 5 0 1）。次いで、有効情報の有無を判断し（S 5 0 2）、ハンドラ内のプログラミング情報が無い、あるいは有効情報をすべて処理し終えた場合には、オブジェクト情報 3 9 を保存し（S 5 0 4）、終了する。

#### 【0 1 9 3】

ステップ S 5 0 2 において、有効情報がある場合には、RTOS カーネルチェック部 3 7 にて、発行システムコールカーネル部を処理し（S 5 0 3）、ステップ S 5 0 1 以降の処理を繰り返す。

#### 【0 1 9 4】

この様な方法により、割り込みイベントを発生させたいオブジェクトの状態を操作し、オブジェクトの状態が割り込み発生できる状態であることをチェック群 4 5 により判定された場合に、タスク部のプログラミング中に、同期がとれてないハンドラ部のプログラミング処理単位を割り込みイベントとして、一つのアプリケーションシステムの動作として表示することができるため、プログラミングの段階において、ハードウェアに発生する周期的な割り込み、非同期な割り込みを検証しながら、作業をすすめることができる。

#### 【0 1 9 5】

##### 〔時間待ち状態における動作〕

次いで、時間待ち状態における動作について説明する。ここでは、アプリケーション開発システム 2 0 において、図 4 0 及び図 4 1 に示すようなプログラミングを入力装置 7 から入力が行われた場合を例に説明する。図 4 0 及び図 4 1 は、本発明装置によるプログラミング中の状態における画面操作を示す説明図である。

#### 【0 1 9 6】

本実施形態では、開発アプリケーションシステムが稼働するハードウェアにお

いて、ハードウェアが実装している時間機能をRTOSが利用し、ある実行状態のタスクを時間待ち状態に遷移させ、プログラミングの段階において、その待ち時間の解除タイミングを定義し、アプリケーションプログラムが使用するタスクの振る舞い、及びオブジェクトの状態変化を検証する方法を示す。

#### 【0197】

図41のタスク部プログラミング作業ペイン111では、目盛り位置：1において、“d l y \_ t s k”（タスク遅延）システムコールが発行されている。“d l y \_ t s k”を発行した実行タスク“\_ u s r 3 c d”は、時間待ち状態へと遷移する。

#### 【0198】

アプリケーションプログラムの実行環境から、時間待ち状態タスク“\_ u s r 3 c d”の待ちを解除する方法は、“r e l \_ w a i”[他タスクの待ち状態解除]を任意の目盛り位置で発行するか、“d l y \_ ? t s k”発行時に指定した遅延時間の時間経過を待つことになる。“r e l \_ w a i”を任意のタイミングで発行し、時間待ち状態を解除する方法は、システムコール発行時の動作で説明した機能により実現できる。

#### 【0199】

もう一つの時間経過を待つ方法は、開発アプリケーションシステムが稼働するハードウェアにおいて、ハードウェアが実装している時間機能を利用しているため、プログラミング段階において、動作を検証することができない。そこで、タイムアウトイベントを任意のタイミングで、実行状態タスクから発行する。

#### 【0200】

タイムアウトイベントの発行は、上述したシステムコールの発行方法と同じである。つまり、タイムアウトを発行するタスク側がシステムコール発行位置であり、操作の対象となるオブジェクトが、時間待ち状態タスクとなる。

#### 【0201】

図41に示す例では、目盛り位置：1で時間待ち状態に遷移し、目盛り位置：9においてタイムアウトイベント131を発行し、“\_ u s r 3 c d”の時間待ち状態を解除している。実施例では、時間待ち状態を解除したことにより、タス



クオブジェクト状態が変化し、目盛り位置：10において、実行状態タスクが”\_usr3cd”になることが検証できている。

#### 【0202】

以上、第3実施例を用いて説明したように、本発明によれば、プログラミング段階において、その実行環境の動作規則に従いアプリケーションシステムの振る舞いを視覚的に確認し、早期にプログラミングの不整合発見を行い、作業行程の戻りを低減させることにより、効率的なアプリケーションシステム開発を可能とすることができる。

#### 【0203】

以上、本発明について、詳細に説明したが、本発明は本実施例に限定されず、本発明の主旨を逸脱しない範囲において、種々の改良や変更を成し得るであろう。例えば、本実施例では、イベントの最後（プログラムの処理終了時点）のタスクの状態に着目して不具合要因を解析する例を示したが、この着目するポイントはこれに限定されず、ユーザーは任意のポイントに着目して不具合要因の解析することができる。

#### 【0204】

また、本実施例ではITRONに準拠したアプリケーション・プログラムを例に説明したが、本発明はこのようなりアルタイムOSに限定されず、一般にマルチタスク方式で動作するプログラムに広く適用できる。

#### 【0205】

##### 【発明の効果】

本発明によれば、対話形式でユーザーから指摘された不具合箇所とプログラムの実行履歴情報から不具合要因とその解決策を特定し、更にこの解決策を反映させたプログラムの動作をユーザーに提示することができる。また、前記特定した不具合要因とその解決策から、該不具合を解決したプログラムを自動生成してユーザーに提示することができる。

#### 【0206】

これらの結果、プログラムの不具合解析作業を容易にし、従来不具合解析に要していた時間を短縮することができ、プログラムの開発コストや開発期間などを

低減することができる。

【 0 2 0 7 】

また、他の発明によれば、プログラミング段階において、その実行環境の動作規則に従いアプリケーションシステムの振る舞いを視覚的に確認し、早期にプログラミングの不整合発見を行い、作業行程の戻りを低減させることにより、効率的なアプリケーションシステム開発を可能とすることができる。

【図面の簡単な説明】

【図 1】

本発明に係る情報処理装置の一実施例を示す概略図である。

【図 2】

本発明に係る情報処理装置に搭載される不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体の処理例を示す流れ図である。

【図 3】

本発明の情報処理装置が利用する検証データのフォーマット例を示す概略図である。

【図 4】

本発明の情報処理装置が利用する比較データのフォーマット例を示す概略図である。

【図 5】

本発明の情報処理装置が利用する判定結果データ 1 のフォーマット例を示す概略図である。

【図 6】

本発明の情報処理装置が利用する判定結果データ 2 のフォーマット例を示す概略図である。

【図 7】

本発明の情報処理装置が利用する不具合解決用質問データのフォーマット例を示す概略図である。

【図 8】

第 1 実施例で想定したアプリケーション・プログラムの仕様に沿った動作状況

を示す入出力関連図である。

【図 9】

第 1 実施例で想定したアプリケーション・プログラムの不具合による動作状況を示す入出力関連図である。

【図 1 0】

図 2 で示した不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体の処理のうち、ユーザーインターフェース部の詳細な処理例を示す流れ図である。

【図 1 1】

図 1 0 で示したプログラムの動作状況をユーザーに提示した表示例示すイメージ図である。

【図 1 2】

図 2 で示した不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体の処理のうち、要求解析部の詳細な処理例を示す流れ図である。

【図 1 3】

図 1 0 で示したプログラムの動作状況をユーザーに提示した表示例示すイメージ図である。

【図 1 4】

図 2 で示した不具合解析プログラムを格納したコンピュータ読み取り可能な記憶媒体の処理のうち、タスク動作判定部の詳細な処理例を示す流れ図である。

【図 1 5】

図 1 0 で示したプログラムの動作状況をユーザーに提示した表示例示すイメージ図である。

【図 1 6】

本発明の情報処理装置が利用するシステムコール対応テーブルの一例を示す概略図である。

【図 1 7】

第 1 実施例における各種データの状態を示した図である。

【図 1 8】

第 1 実施例における各種データの状態を示した図である。

【図 1 9】

第 2 実施例で想定したアプリケーション・プログラムの仕様に沿った動作状況を示す入出力関連図である。

【図 2 0】

第 1 実施例で想定したアプリケーション・プログラムの不具合による動作状況を示す入出力関連図である。

【図 2 1】

第 2 実施例における各種データの状態を示した図である。

【図 2 2】

第 2 実施例における各種データの状態を示した図である。

【図 2 3】

第 2 実施例における各種データの状態を示した図である。

【図 2 4】

第 2 実施例における各種データの状態を示した図である。

【図 2 5】

第 2 実施例における各種データの状態を示した図である。

【図 2 6】

第 2 実施例における各種データの状態を示した図である。

【図 2 7】

第 3 実施形態におけるハードウェア資源の構成を示すブロック図である。

【図 2 8】

第 3 実施形態におけるアプリケーションプログラム開発システムの全体構成を示すブロック図である。

【図 2 9】

第 3 実施形態に係るプログラミング装置 3 2 の G U I の画面構成を示す説明図である。

【図 3 0】

第 3 実施形態に係るオブジェクト表示ペインの画面構成を示す説明図である。

【図 3 1】

第 3 実施形態において、開発装置を利用しアプリケーションシステムを開発する流れを示すフロー図である。

【図 3 2】

第 3 実施形態において、プログラミング装置 3 2 を利用し、システムコールを発行した時の、開発装置の処理の流れを示すフロー図である。

【図 3 3】

第 3 実施形態において、システムコールを発行する場合の画面操作を示す説明図である。

【図 3 4】

第 3 実施形態において、システムコール発行位置チェックの流れを示すフロー図である。

【図 3 5】

第 3 実施形態において、システムコール操作対象位置チェックの流れを示すフロー図である。

【図 3 6】

第 3 実施形態における実行状態ナビゲートラインの説明図である。

【図 3 7】

第 3 実施形態におけるタスク部プログラミング作業ペインとハンドラ部プログラミング作業ペインの画面操作を示す説明図である。

【図 3 8】

第 3 実施形態における割り込み発生イベントをプログラミング中に仮想定義する際の画面操作を示す説明図である。

【図 3 9】

第 3 実施形態における割り込み発生イベント発行時の流れを示すフロー図である。

【図 4 0】

第 3 実施形態における割り込み発生イベント定義時のアプリケーションプログラムの処理操作を示す説明図である。

【図 4 1】

第 3 実施形態において、時間経過をイベントとし、プログラミング中に仮想定義する操作を示す説明図である。

【図 4 2】

従来のデバッガの構成を示した概略図である。

【図 4 3】

従来のデバッガが出力する実行履歴情報の構成を示した概略図である。

【図 4 4】

従来の実行履歴情報の構成とデータの一例を示したイメージ図である。

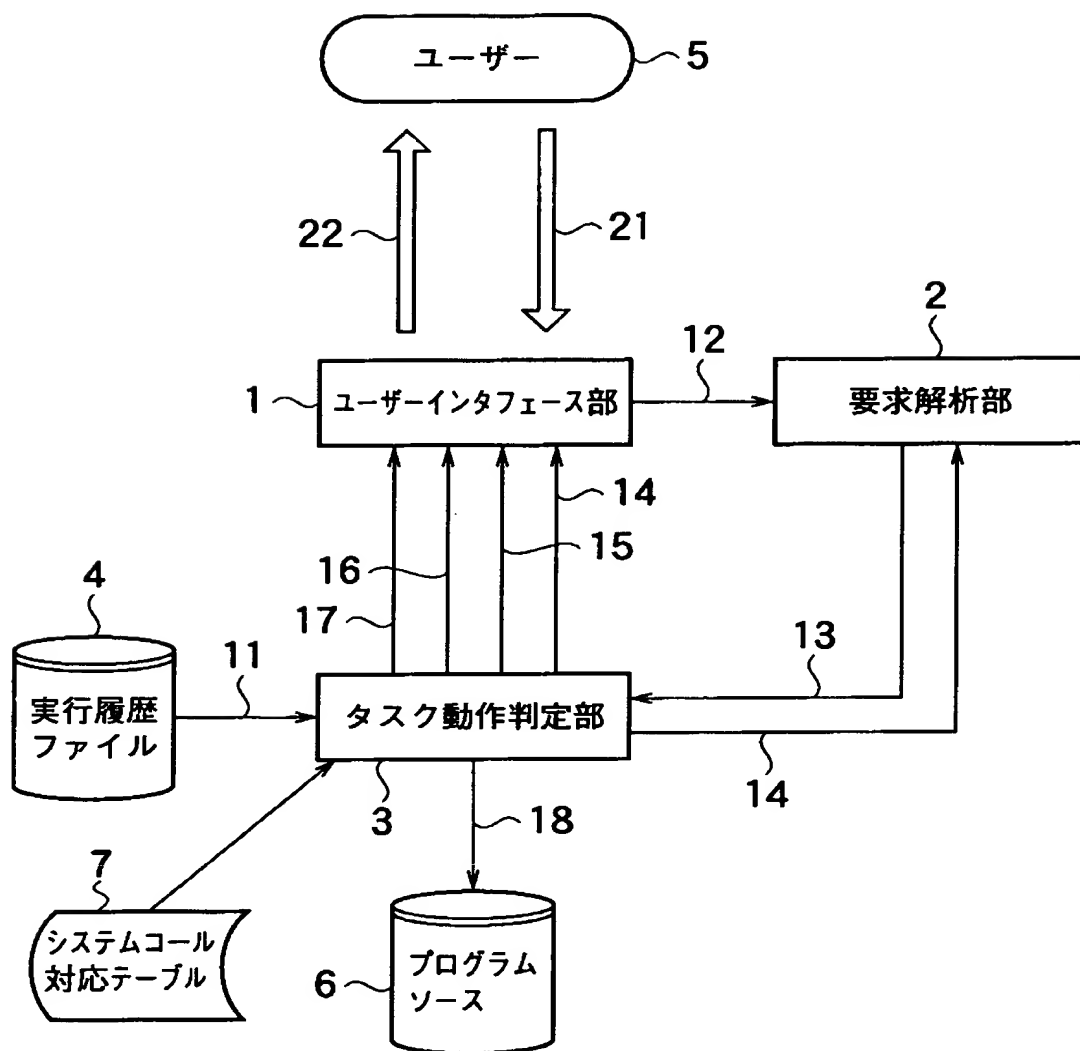
【符号の説明】

- 1 .... ユーザーインタフェース部
- 2 .... 要求解析部
- 3 .... タスク動作判定部
- 4 .... 実行履歴ファイル
- 5 .... ユーザー
- 6 .... システムコール対応テーブル
- 1 1 .... 実行履歴情報
- 1 2 .... 入力データ
- 1 3 .... 比較データ
- 1 4 .... 検証データ
- 1 5 .... 判定結果データ 1
- 1 6 .... 判定結果データ 2
- 1 7 .... 不具合質問用データ
- 1 8 .... プログラム
- 1 0 1 .... OS
- 1 0 2 .... デバッガ
- 5 1 ... 中央処理装置
- 5 2 ... RAM
- 5 3 ... ROM

- 5 4 …通信装置
- 5 5 …補助記憶装置
- 5 6 …表示装置
- 5 7 …入力装置
- 5 8 …バス
- 5 9 …出力装置
- 3 1 …環境定義部
- 3 2 …プログラミング部
- 4 0 …テンプレートファイル生成部
- 4 5 …チェック部
- 4 6 …実行オブジェクト生成部
- 4 8 …表示情報生成部

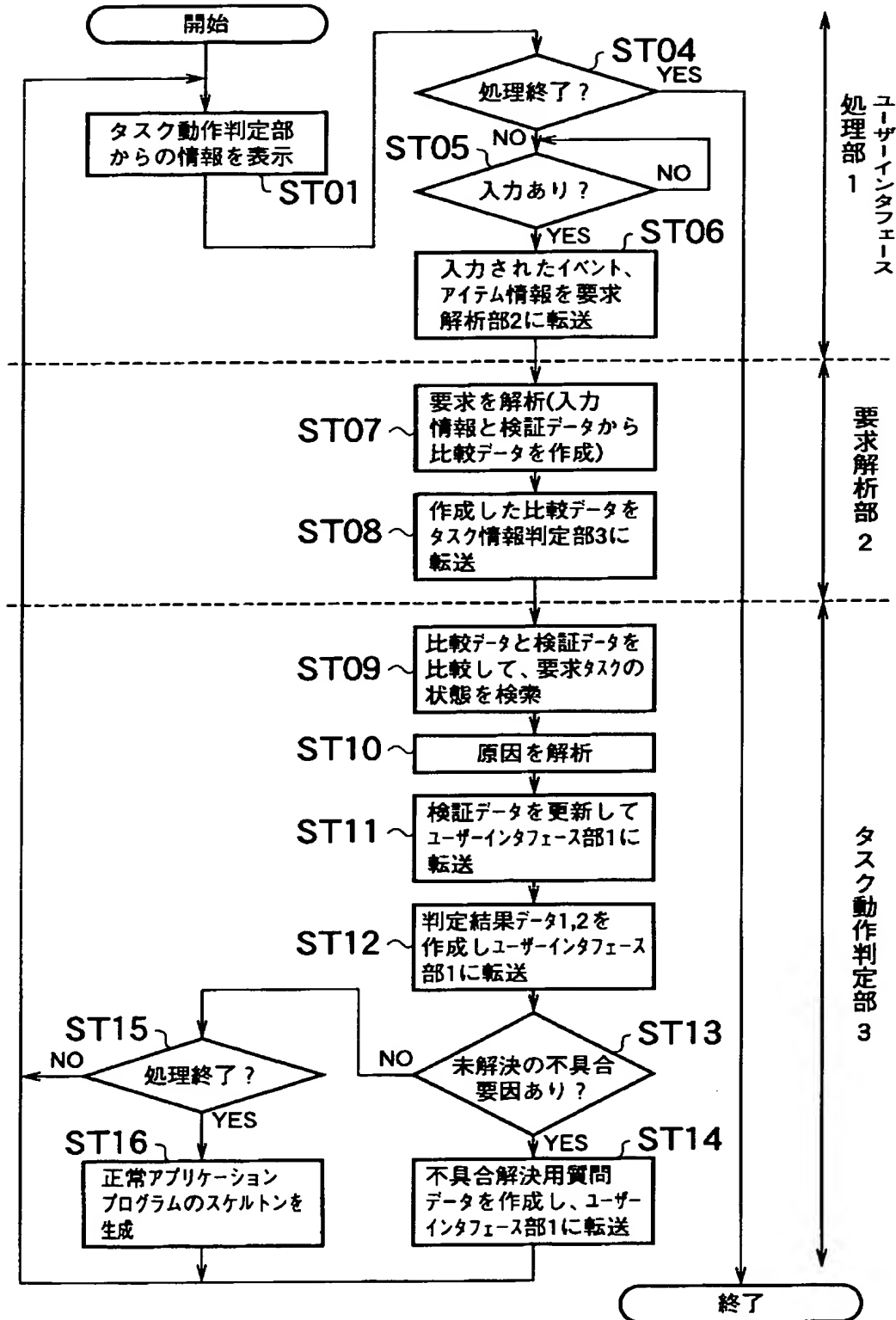
【書類名】 図面

【図 1】





【図 2】



【図 3】

イベント順	イベント属性 (システムコールの発行)	発行システムコール	発行元タスクID	発行元タスク優先度	発行元タスク後のタスク状態	発行先タスクID(発行先資源)	発行先タスク優先度(発行先ID)	発行先タスク後のタスク状態
イベント順	イベント属性 (ディスパッチ)	ディスパッチ元タスクID	ディスパッチ元タスク優先度	ディスパッチ元タスクID	ディスパッチ先タスク優先度	-	-	-
イベント順	イベント属性 (割り込み処理)	ハンドラ属性 (周期起動 ハンドラ、アラーム ハンドラ、割り込み ハンドラ)	ハンドラNo	-	-	-	-	-
・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・

イベント属性：システムコールの発行  
ディスパッチ割り込み処理

ハンドラ属性：周期起動ハンドラ  
アラームハンドラ  
割り込みハンドラ

【図 4】

N	先イベント	後イベント	アイテム(タスク (y))
	:	:	:

⏟  
タイミング (x)

【図 5】

① N	② 要求タスク	③ 要求タスクの状態
:	:	:
:	:	:

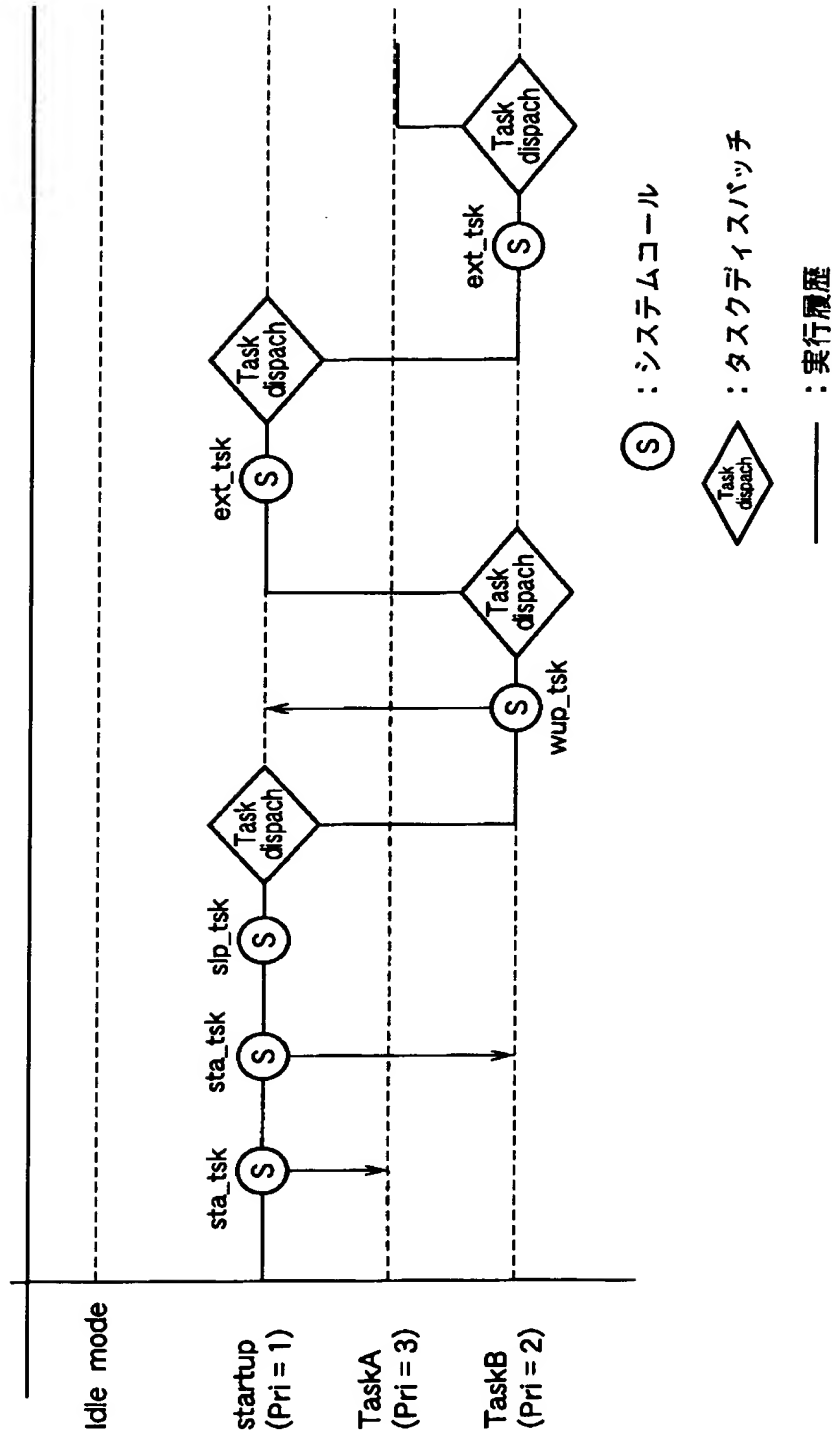
【図 6】

① N	② イベントの最後に実行状態のタスク	③ ext_tsk
:	:	:
:	:	:

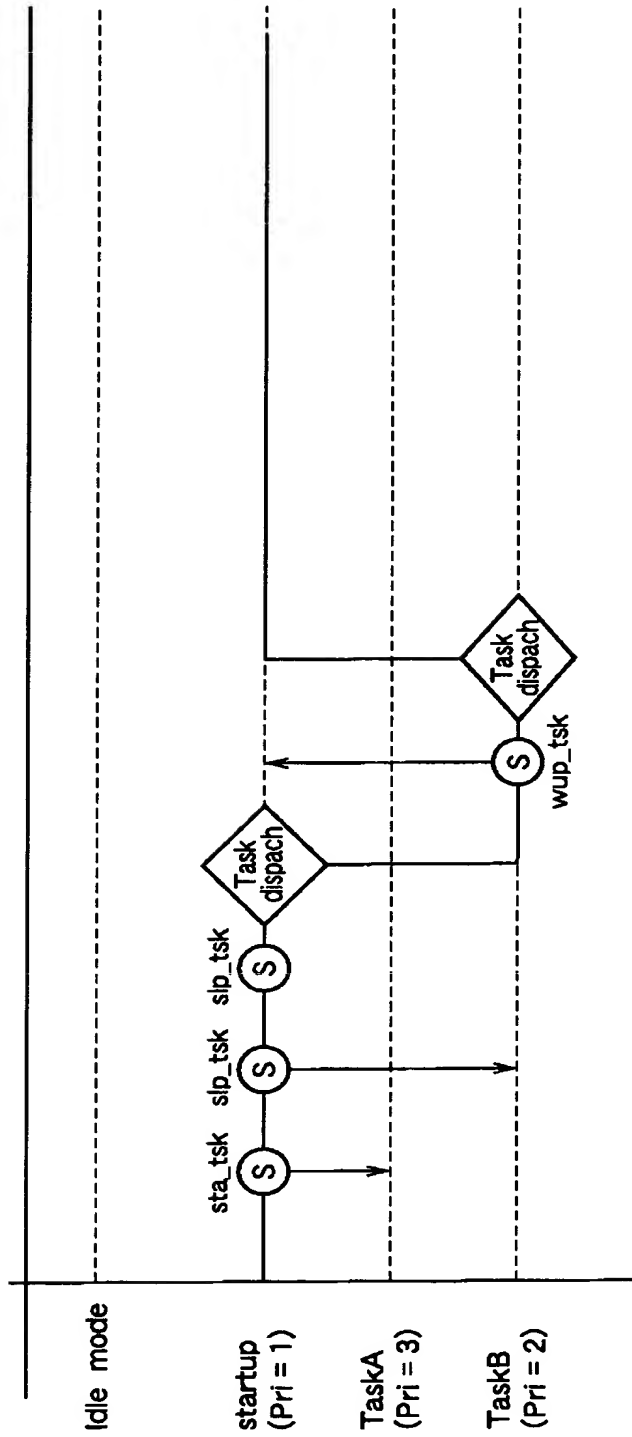
【図 7】

① N	② 要求タスクを実行可能状態にする システムコールの発行対象	③ 要求タスクを実行可能状態にする システムコール
:	:	:
:	:	:

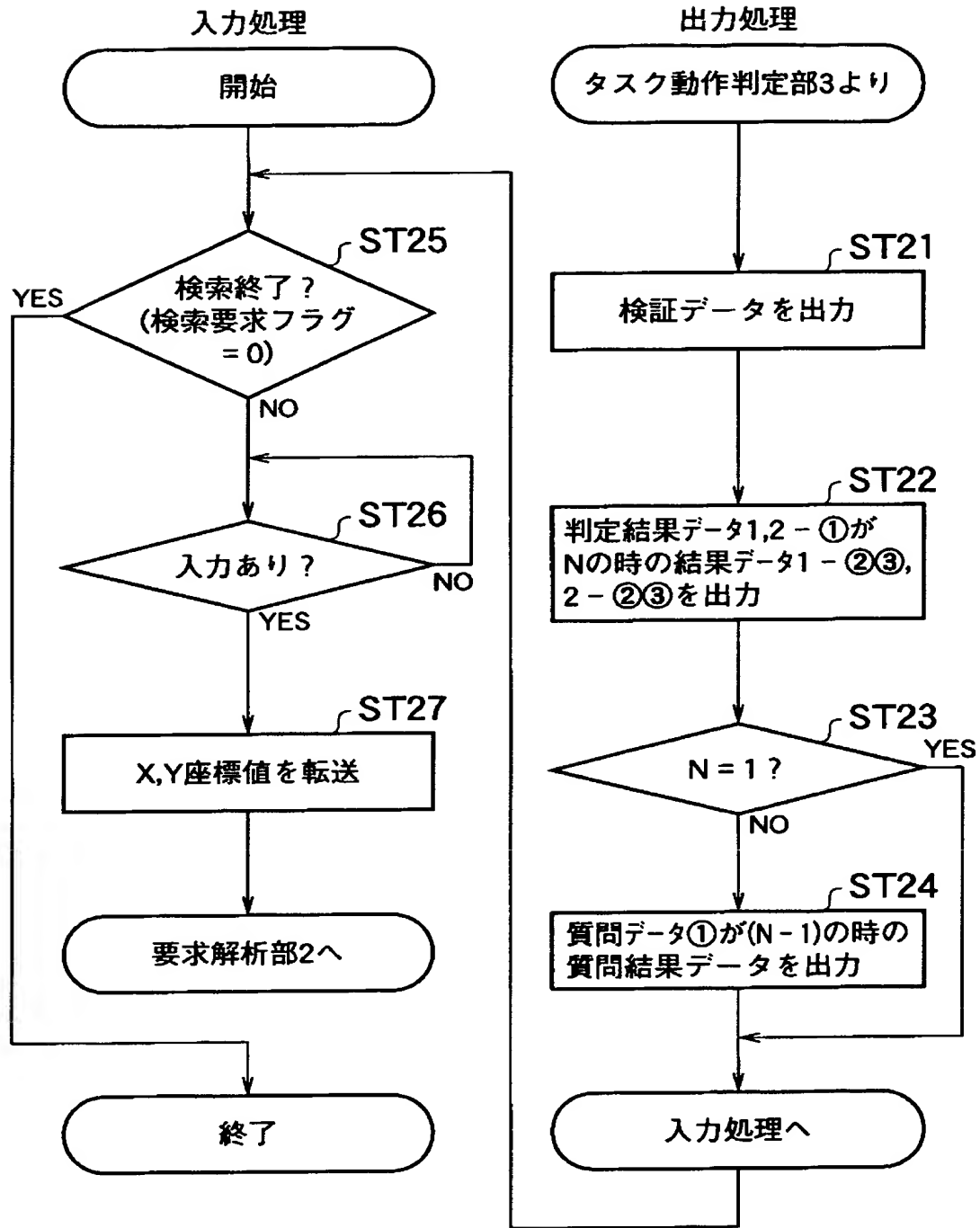
【図 8】



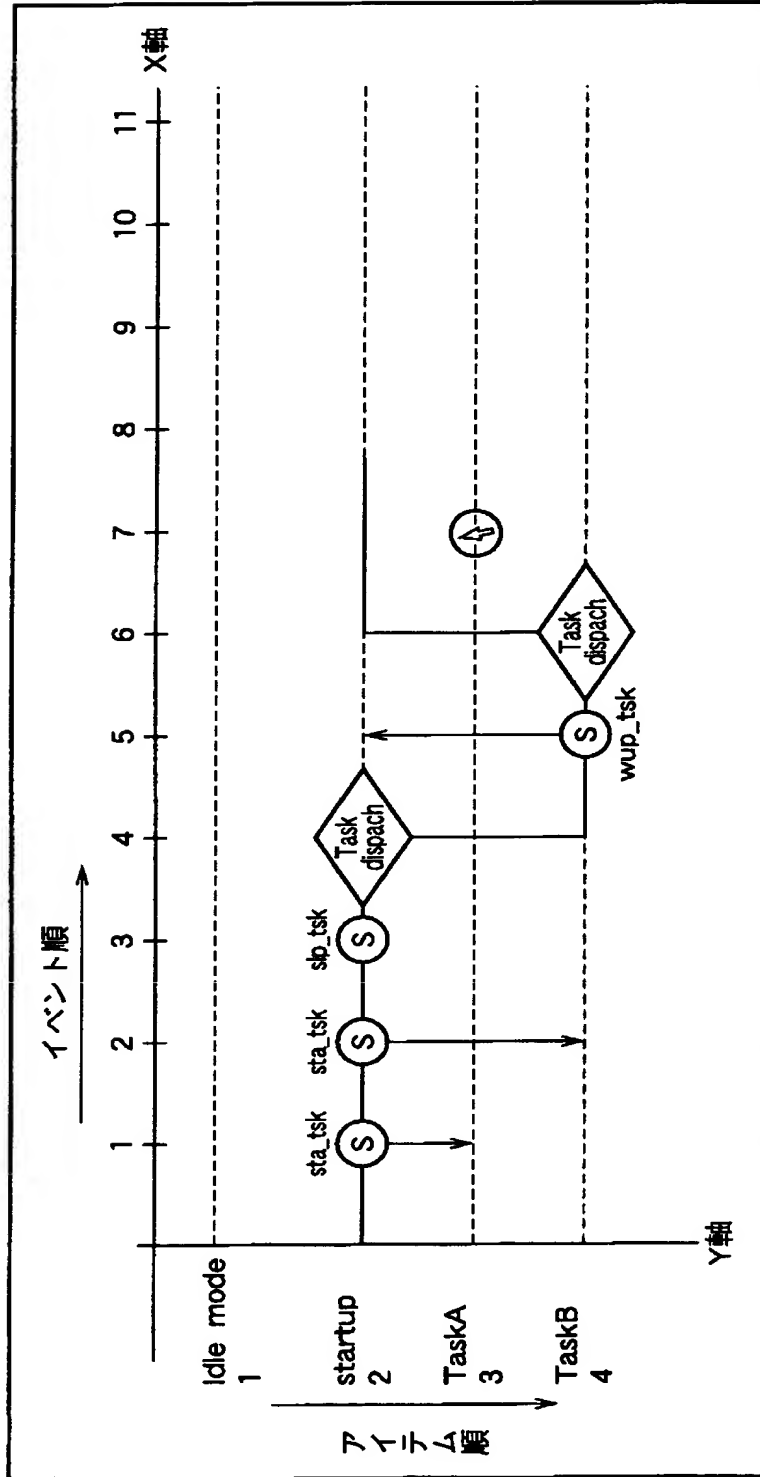
【图 9】



【図10】



【図 11】



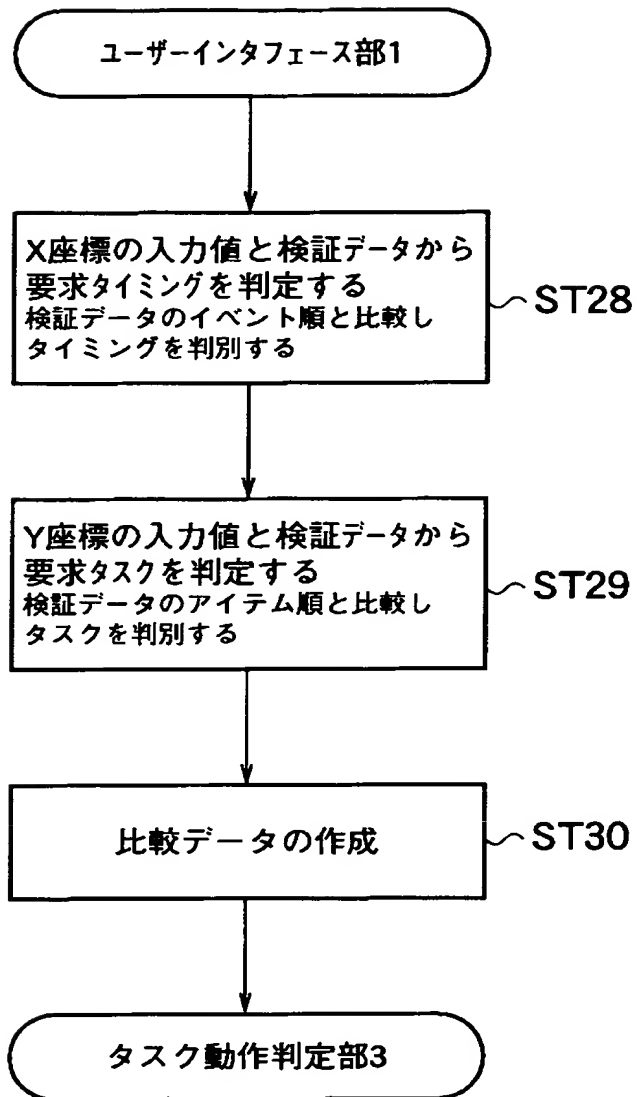
入力条件：実行履歴を表す線上升しか指定できない

マウスカーソル

マウスカーソルをシステムコールを発行したい場所に近づける

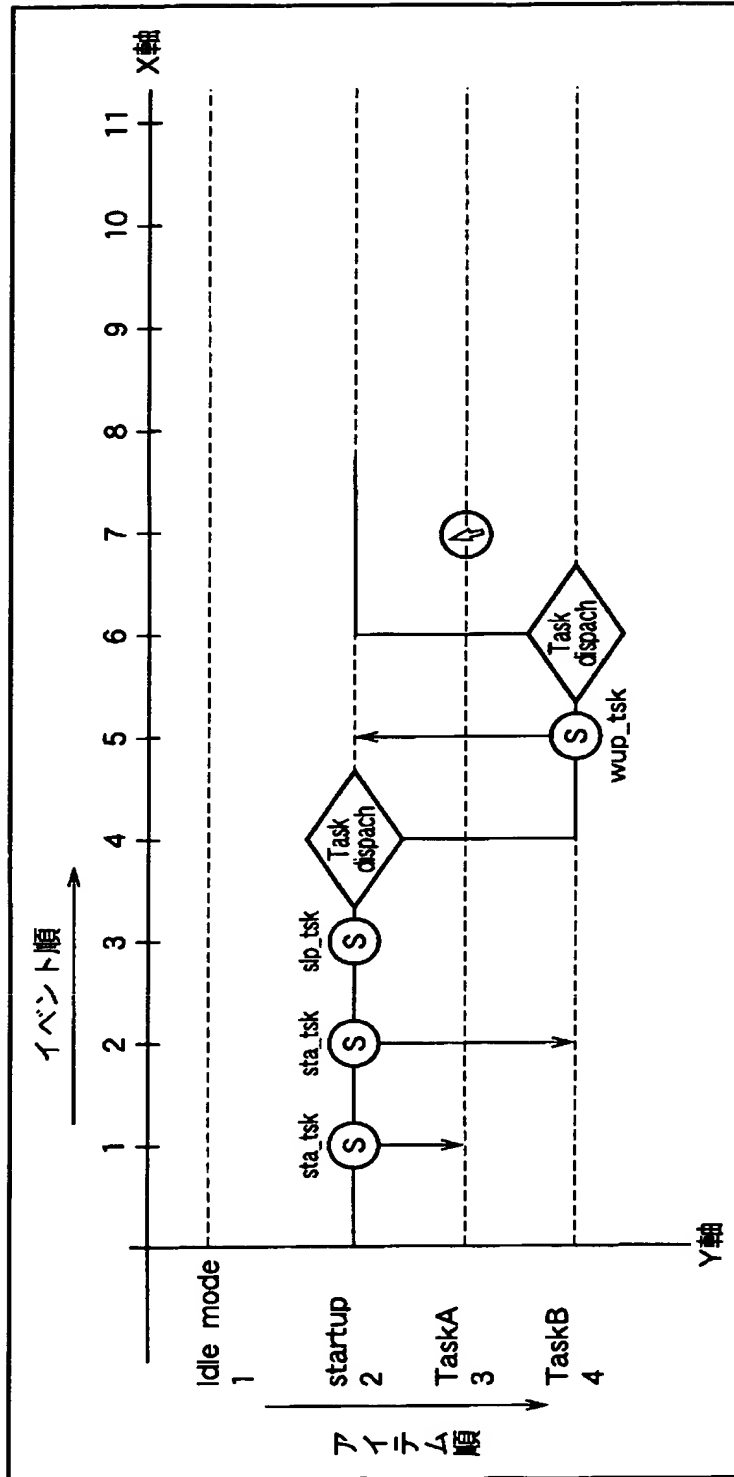
X座標	7
Y座標	3

【図 1 2】





【図 1 3】



マウスカーソルをシステムコールを発行したい場所に近づけた場合

比較データ

X座標	7
Y座標	3

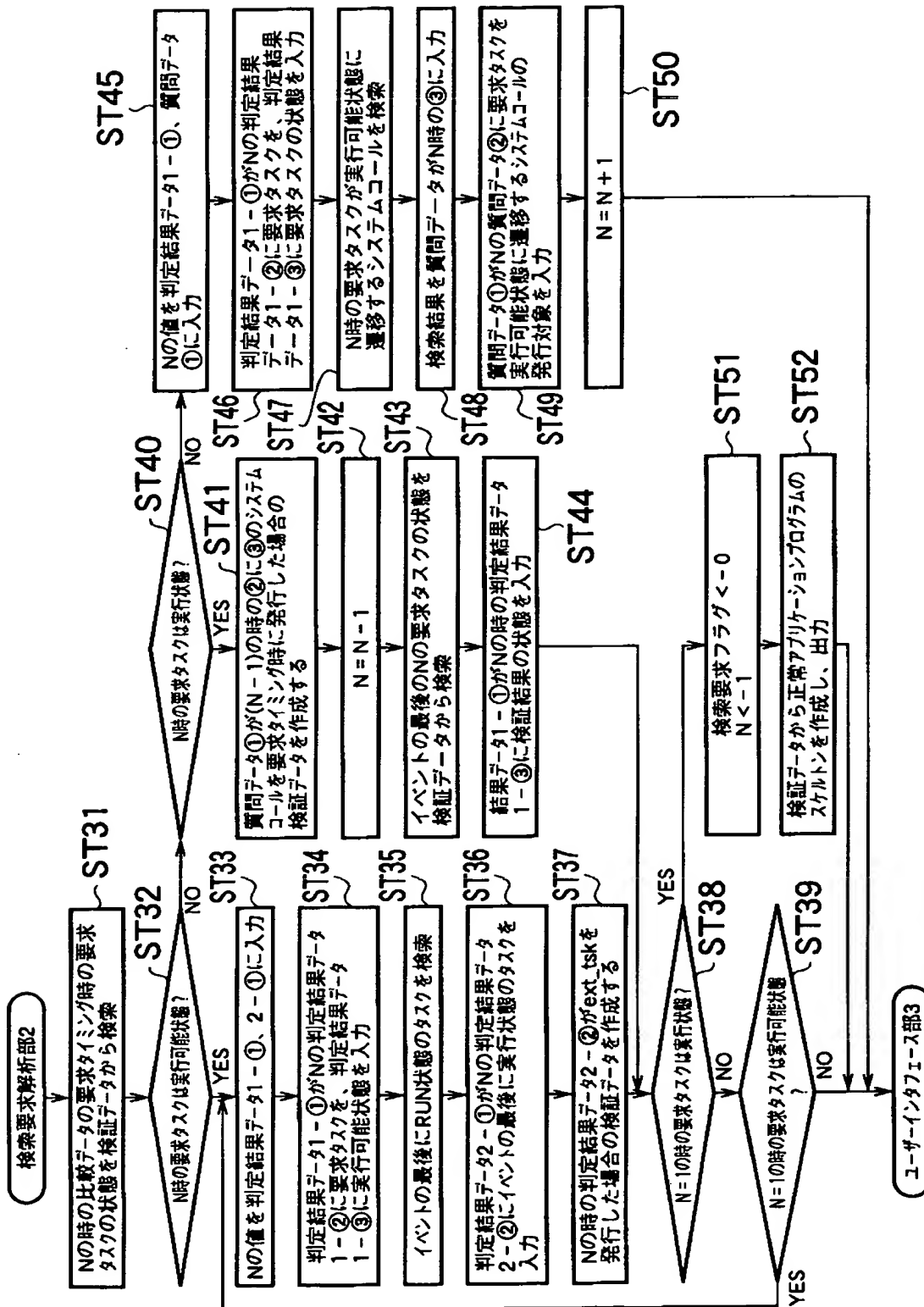
(1次入力)

1	7	7	7	タスクA
---	---	---	---	------

要求タイミング

要求タスク

【図 14】



【図 1 5】

<システムコール対応テーブル>

No	システムコール	対応するシステムコール
1	sta_tsk	ext_tsk
2	slp_tsk	wup_tsk
3	wai_sem	sig_sem
⋮	⋮	⋮

【図 1 6】

● [処理1] → [第1次要求]

<検証データ>

イベント順	イベント属性	発行システムコール	ハンドラ属性	ハンドラNO	発行元タスクID	発行元タスク優先度	発行元タスクの発行後のタスク状態	発行先タスクID(発行先資源)	発行先タスク優先度(発行先ID)	発行先タスクの発行後のタスク状態
1	System Call	sta_tsk	-	-	startup	1	実行中	taskA	3	実行可能
2	System Call	sta_tsk	-	-	startup	1	実行中	taskB	2	実行可能
3	System Call	slp_tsk	-	-	startup	1	待ち	-	-	-
4	Task Dispatch	-	-	-	startup	1	-	taskB	2	実行中
5	System Call	wup_tsk	-	-	taskB	2	実行可能	startup	1	実行可能
6	Task Dispatch	-	-	-	taskB	2	-	startup	1	実行中

<比較データ>

N	先イベント	後イベント	アイテム(タスク(y))
1	7	7	taskA ★

<判定結果データ1>

① N	② 要求タスク	③ 要求タスクの状態

<判定結果データ2>

① N	② イベントの最終に実行状態のタスク	③ ext_tsk

<不具合解決用質問データ>

① N	② 要求タスクを実行可能にするシステムコールの発行対象	③ 要求タスクを実行可能状態にするシステムコール

【図 1 7】

●【処理2】→【処理終了】

<検証データ>

イベント順	イベント属性	発行システムコール	ハンドラ属性	ハンドラNO	発行元タスクID	発行元タスク優先度	新タスクの発行後のタスク状態	発行先タスクID(発行先資源)	発行先タスク優先度(発行先ID)	発行先タスクの発行後のタスク状態
1	System Call	sta_tsk	-	-	startup	1	実行中	taskA	3	実行可能
2	System Call	sta_tsk	-	-	startup	1	実行中	taskB	2	実行可能
3	System Call	slp_tsk	-	-	startup	1	待ち	-	-	-
4	Task Dispatch	-	-	-	startup	1	-	taskB	2	実行中
5	System Call	wup_tsk	-	-	taskB	2	実行中	startup	1	実行可能
6	Task Dispatch	-	-	-	taskB	2	-	startup	1	実行中
7	System Call	ext_tsk	-	-	startup	1	休止	-	-	-
8	Task Dispatch	-	-	-	startup	1	-	taskB	2	実行状態
9	System Call	ext_tsk	-	-	taskB	2	休止	-	-	-
10	Task Dispatch	-	-	-	taskB	2	-	taskA	3	実行状態

★  
★  
★  
★

<比較データ>

N	先イベント	後イベント	アイテム(タスク(y))
1	7	7	taskA

<判定結果データ1>

① N	② 要求タスク	③ 要求タスクの状態
1	taskA	実行可能 ★
1	taskA	実行可能 ★

<判定結果データ2>

① N	④ イベントの最後に実行状態のタスク	⑤ ext_tsk
1	startup	ext_tsk ★
1	taskB	ext_tsk ★

<不具合解決用質問データ>

① N	⑥ 要求タスクを実行可能にするシステムコールの発行対象	⑦ 要求タスクを実行可能状態にするシステムコール

【図 1 8】

## 実施例 1 正常アプリケーションのスケルトン例

```

/*****
File      : sample.c
Data      : 1999/11/11
Developer : TOSHIBA
Application Skeleton
*****/
#include "itron.h"

#define TASK_ID1      1
#define TASK_ID2      2
#define TASK_ID3      3

TASK startup() :
TASK TaskA() :
TASK TaskB() :

TASK startup() :
{
    ER ercd ;

    ercd = sta_tsk(TASK_ID2,0) ;
    ercd = sta_tsk(TASK_ID3,0) ;
    ercd = slp_tsk() ;

    ext_tsk() ; ----- (a)
}

TASK TaskA()
{
    for( ; ; ){
    }

}

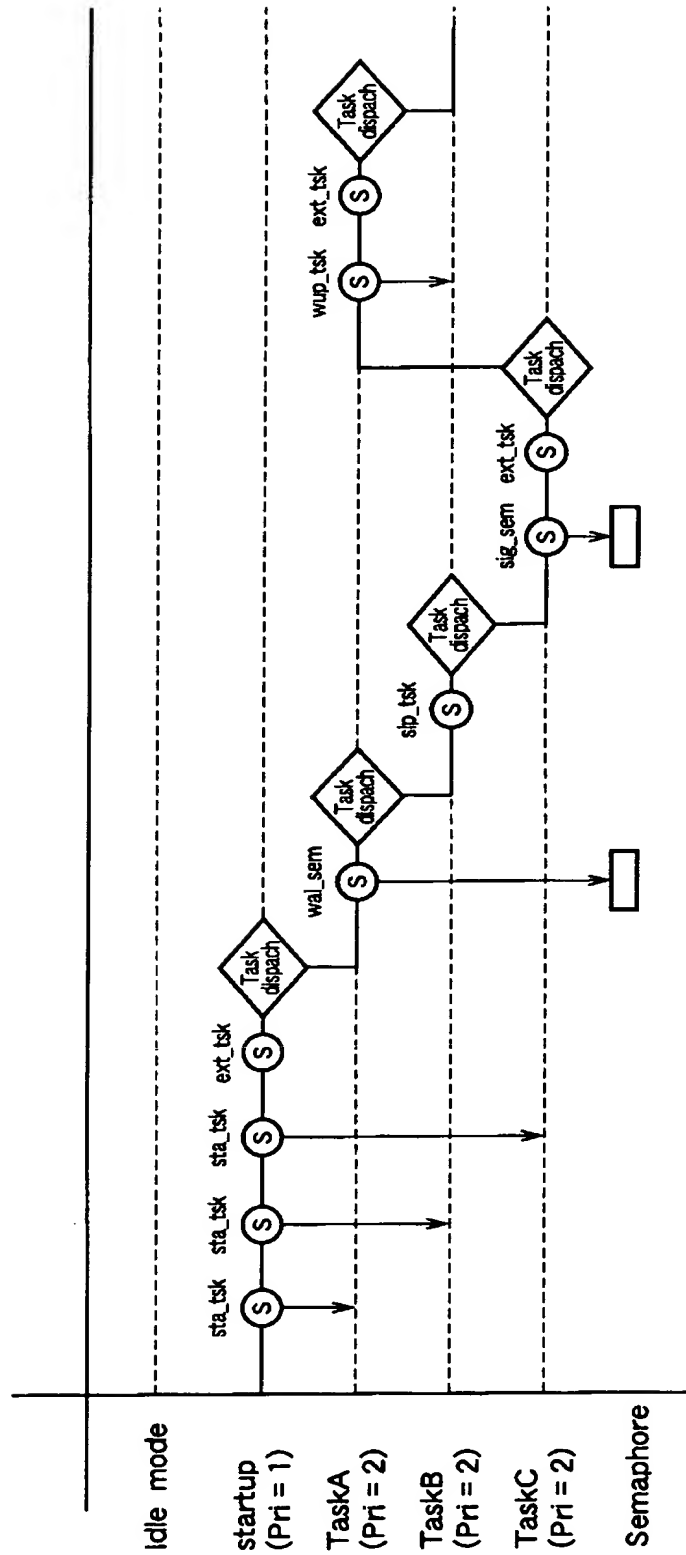
TASK TaskB()
{
    ER ercd ;

    ercd = wup_tsk(TASK_ID1) ;

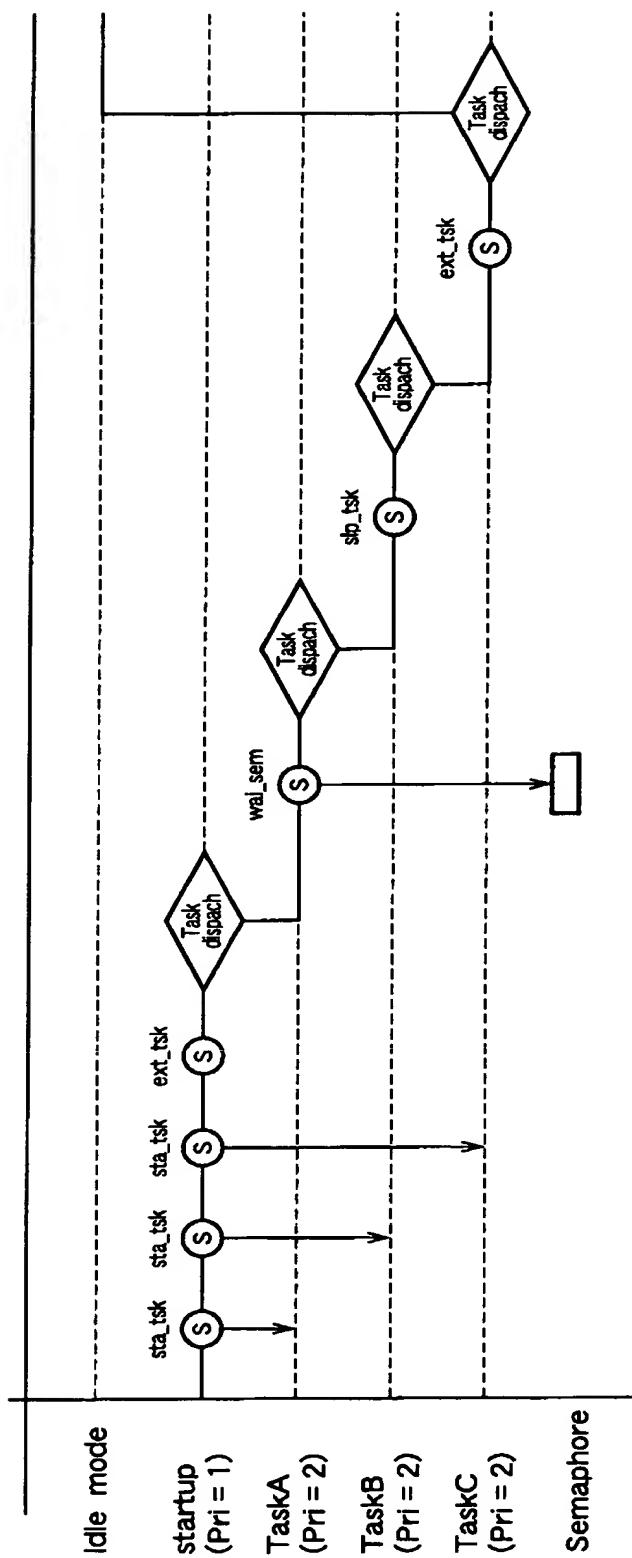
    ext_tsk() ; ----- (b)
}

```

【図 19】



【圖 20】





【図 2 1】

## ●【処理1】→【第1次要求】

## &lt;検証データ&gt;

イベント順	イベント属性	発行システムコール	ハンドラ属性	ハンドラNO	発行元タスクID	発行元タスク優先度	発行元タスクの発行後のタスク状態	発行先タスクID(発行先資源)	発行先タスク優先度(発行先ID)	発行先タスクの発行後のタスク状態
1	System Call	sta_tsk	-	-	startup	1	実行状態	taskA	2	実行可能
2	System Call	sta_tsk	-	-	startup	1	実行状態	taskB	2	実行可能
3	System Call	sta_tsk	-	-	startup	1	実行状態	taskC	2	実行可能
4	System Call	ext_tsk	-	-	startup	1	休止	-	-	-
5	Task Dispatch	-	-	-	startup	1	-	taskA	2	実行状態
6	System Call	wai_sem	-	-	taskA	2	待ち	Semaphore	1	-
7	Task Dispatch	-	-	-	taskA	2	-	taskB	2	実行状態
8	System Call	slp_tsk	-	-	taskB	2	待ち	-	-	-
9	Task Dispatch	-	-	-	taskB	2	-	taskC	2	実行状態
10	System Call	ext_tsk	-	-	taskC	2	休止	-	-	-
11	Task Dispatch	-	-	-	taskC	2	-	idle Mode	-	-

## &lt;比較データ&gt;

N	先イベント	後イベント	アイテム(タスク(y))
1	12	12	taskB ★

## &lt;判定結果データ1&gt;

① N	② 要求タスク	③ 要求タスクの状態

## &lt;判定結果データ2&gt;

① N	② イベントの最後に実行状態のタスク	③ ext_tsk

## &lt;不具合解決用質問データ&gt;

① N	② 要求タスクを実行可能にするシステムコールの発行対象	③ 要求タスクを実行可能状態にするシステムコール

【図 2 2】

● [処理2] → [第2次要求]

<検証データ>

イベント順	イベント属性	発行システムコール	ハンドラ属性	ハンドラNO	発行元タスクID	発行元タスク優先度	発行元タスクの発行後のタスク状態	発行先タスクID(発行先資源)	発行先タスク優先度(発行先ID)	発行先タスクの発行後のタスク状態
1	System Call	sta_tsk	-	-	startup	1	実行状態	taskA	2	実行可能
2	System Call	sta_tsk	-	-	startup	1	実行状態	taskB	2	実行可能
3	System Call	sta_tsk	-	-	startup	1	実行状態	taskC	2	実行可能
4	System Call	ext_tsk	-	-	startup	1	休止	-	-	-
5	Task Dispatch	-	-	-	startup	1	-	taskA	2	実行状態
6	System Call	wai_sem	-	-	taskA	2	待ち	Semaphore	1	-
7	Task Dispatch	-	-	-	taskA	2	-	taskB	2	実行状態
8	System Call	slp_tsk	-	-	taskB	2	待ち	-	-	-
9	Task Dispatch	-	-	-	taskB	2	-	taskC	2	実行状態
10	System Call	ext_tsk	-	-	taskC	2	休止	-	-	-
11	Task Dispatch	-	-	-	taskC	2	-	Idle Mode	-	-

<比較データ>

N	先イベント	後イベント	アイテム(タスク(y))
1	12	12	taskB
2	6	7	taskA
			★

<判定結果データ1>

① N	② 要求タスク	③ 要求タスクの状態
1	taskB	待ち
		★

<判定結果データ2>

① N	② イベントの最後に実行状態のタスク	③ ext_tsk

<不具合解決用質問データ>

① N	② 要求タスクを実行可能にするシステムコールの発行対象	③ 要求タスクを実行可能状態にするシステムコール
1	taskA	wup_tsk
		★

【図 2 3】

## ●【処理3】→【第3次要求】

## &lt;検証データ&gt;

イベント順	イベント属性	発行システムコール	ハンドラ属性	ハンドラNO	発行元タスクID	発行元タスク優先度	発行元タスクの発行後のタスク状態	発行先タスクID(発行先資源)	発行先タスク優先度(発行先ID)	発行先タスクの発行後のタスク状態
1	System Call	sta_tsk	-	-	startup	1	実行状態	taskA	2	実行可能
2	System Call	sta_tsk	-	-	startup	1	実行状態	taskB	2	実行可能
3	System Call	sta_tsk	-	-	startup	1	実行状態	taskC	2	実行可能
4	System Call	ext_tsk	-	-	startup	1	休止	-	-	-
5	Task Dispatch	-	-	-	startup	1	-	taskA	2	実行状態
6	System Call	wai_sem	-	-	taskA	2	待ち	Semaphore	1	-
7	Task Dispatch	-	-	-	taskA	2	-	taskB	2	実行状態
8	System Call	slp_tsk	-	-	taskB	2	待ち	-	-	-
9	Task Dispatch	-	-	-	taskB	2	-	taskC	-	実行状態
10	System Call	ext_tsk	-	-	taskC	2	休止	-	-	-
11	Task Dispatch	-	-	-	taskC	2	-	Idle Mode	-	-

## &lt;比較データ&gt;

N	先イベント	後イベント	アイテム(タスク(y))
1	12	12	taskB
2	6	7	taskA
3	9	10	taskC

★

## &lt;判定結果データ1&gt;

① N	② 要求タスク	③ 要求タスクの状態
1	taskB	待ち
2	taskA	待ち

★

## &lt;判定結果データ2&gt;

① N	② イベントの最後に実行状態のタスク	③ ext_tsk

## &lt;不具合解決用質問データ&gt;

① N	② 要求タスクを実行可能にするシステムコールの発行対象	③ 要求タスクを実行可能状態にするシステムコール
1	taskA	wup_tsk
2	taskC	sig_sem

★

【図 2 4】

● [処理4] → [第4次要求]

<検証データ>

イベント順	イベント属性	発行システムコール	ハンドラ属性	ハンドラNO	発行元タスクID	発行元タスク優先度	発行元タスクの発行後のタスク状態	発行先タスクID(発行先資源)	発行先タスク優先度(発行先ID)	発行先タスクの発行後のタスク状態
1	System Call	sta_tsk	-	-	startup	1	実行状態	taskA	2	実行可能
2	System Call	sta_tsk	-	-	startup	1	実行状態	taskB	2	実行可能
3	System Call	sta_tsk	-	-	startup	1	実行状態	taskC	2	実行可能
4	System Call	ext_tsk	-	-	startup	1	休止	-	-	-
5	Task Dispatch	-	-	-	startup	1	-	taskA	2	実行状態
6	System Call	wai_sem	-	-	taskA	2	待ち	Semaphore	1	-
7	Task Dispatch	-	-	-	taskA	2	-	taskB	2	実行状態
8	System Call	slp_tsk	-	-	taskB	2	待ち	-	-	-
9	Task Dispatch	-	-	-	taskB	2	-	taskC	2	実行状態
10	System Call	sig_sem	-	-	taskC	2	実行状態	Semaphore	1	-
11	System Call	ext_tsk	-	-	taskC	2	休止	-	-	-
12	Task Dispatch	-	-	-	taskC	2	-	taskA	2	実行状態

<比較データ>

N	先イベント	後イベント	アイテム(タスク(y))
1	12	12	taskB
2	6	7	taskA
3	9	10	taskC

<判定結果データ1>

① N	② 要求タスク	③ 要求タスクの状態
1	taskB	待ち
2	taskA	実行状態

<判定結果データ2>

① N	② イベントの最後に実行状態のタスク	③ ext_tsk

<不具合解決用質問データ>

① N	② 要求タスクを実行可能にするシステムコールの発行対象	③ 要求タスクを実行可能状態にするシステムコール
1	taskA	wup_tsk
2	taskC	sig_sem

【図 25】

●【処理5】→【第5次要求】

<検証データ>

イベント順	イベント属性	発行システムコール	ハンドラ属性	ハンドラNO	発行元タスクID	発行元タスク優先度	発行元タスクの発行後のタスク状態	発行先タスクID(発行先資源)	発行先タスクの優先度(発行先ID)	発行先タスクの発行後のタスク状態
1	System Call	sta_tsk	-	-	startup	1	実行状態	taskA	2	実行可能
2	System Call	sta_tsk	-	-	startup	1	実行状態	taskB	2	実行可能
3	System Call	sta_tsk	-	-	startup	1	実行状態	taskC	2	実行可能
4	System Call	ext_tsk	-	-	startup	1	休止	-	-	-
5	Task Dispatch	-	-	-	startup	1	-	taskA	2	実行状態
6	System Call	wai_sem	-	-	taskA	2	待ち	Semaphore	1	-
7	Task Dispatch	-	-	-	taskA	2	-	taskB	2	実行状態
8	System Call	slp_tsk	-	-	taskB	2	待ち	-	-	-
9	Task Dispatch	-	-	-	taskB	2	-	taskC	2	実行状態
10	System Call	sig_sem	-	-	taskC	2	実行状態	Semaphore	1	-
11	System Call	ext_tsk	-	-	taskC	2	休止	-	-	-
12	Task Dispatch	-	-	-	taskC	2	-	taskA	2	実行状態
13	System Call	wup_tsk	-	-	taskA	2	実行状態	taskB	2	実行可能★
14	System Call	ext_tsk	-	-	taskA	2	休止	-	-	-
15	Task Dispatch	-	-	-	taskA	2	-	taskB	2	実行状態

<比較データ>

N	先イベント	後イベント	アイテム(タスク(y))
1	12	12	taskB
2	6	7	taskA
3	9	10	taskC
1	13	13	taskB★
2	11	12	taskA

<判定結果データ1>

① N	② 要求タスク	③ 要求タスクの状態
1	taskB	実行状態★
2	taskA	実行状態

<判定結果データ2>

① N	② イベントの最後に実行状態のタスク	③ ext_tsk
1	taskA	ext_tsk★

<不具合解決用質問データ>

① N	② 要求タスクを実行可能にするシステムコールの発行対象	③ 要求タスクを実行可能状態にするシステムコール
1	taskA	wup_tsk
2	taskC	sig_sem

【図 26】

● [処理6] → [処理終了]

<検証データ>

イベント順	イベント属性	発行システムコール	ハンドラ属性	ハンドラNO	発行元タスクID	発行元タスク優先度	発行元タスクの発行後のタスク状態	発行先タスクID(発行先資源)	発行先タスク優先度(発行先ID)	発行先タスクの発行後のタスク状態
1	System Call	sta_tsk	-	-	startup	1	実行状態	taskA	2	実行可能
2	System Call	sta_tsk	-	-	startup	1	実行状態	taskB	2	実行可能
3	System Call	sta_tsk	-	-	startup	1	実行状態	taskC	2	実行可能
4	System Call	ext_tsk	-	-	startup	1	休止	-	-	-
5	Task Dispatch	-	-	-	startup	1	-	taskA	2	実行状態
6	System Call	wai_sem	-	-	taskA	2	待ち	Semaphore	1	-
7	Task Dispatch	-	-	-	taskA	2	-	taskB	2	実行状態
8	System Call	slp_tsk	-	-	taskB	2	待ち	-	-	-
9	Task Dispatch	-	-	-	taskB	2	-	taskC	2	実行状態
10	System Call	sig_sem	-	-	taskC	2	実行状態	Semaphore	1	-
11	System Call	ext_tsk	-	-	taskC	2	休止	-	-	-
12	Task Dispatch	-	-	-	taskC	2	-	taskA	2	実行状態
13	System Call	wup_tsk	-	-	taskA	2	実行状態	taskB	2	実行可能
14	System Call	ext_tsk	-	-	taskA	2	休止	-	-	-
15	Task Dispatch	-	-	-	taskA	2	-	taskB	2	実行状態

★

★

<比較データ>

N	先イベント	後イベント	アイテム(タスク(y))
1	12	12	taskB
2	6	7	taskA
3	9	10	taskC
1	13	13	taskB
2	6	7	taskA

<判定結果データ1>

① N	② 要求タスク	③ 要求タスクの状態
1	taskB	実行可能 ★
2	taskA	実行状態

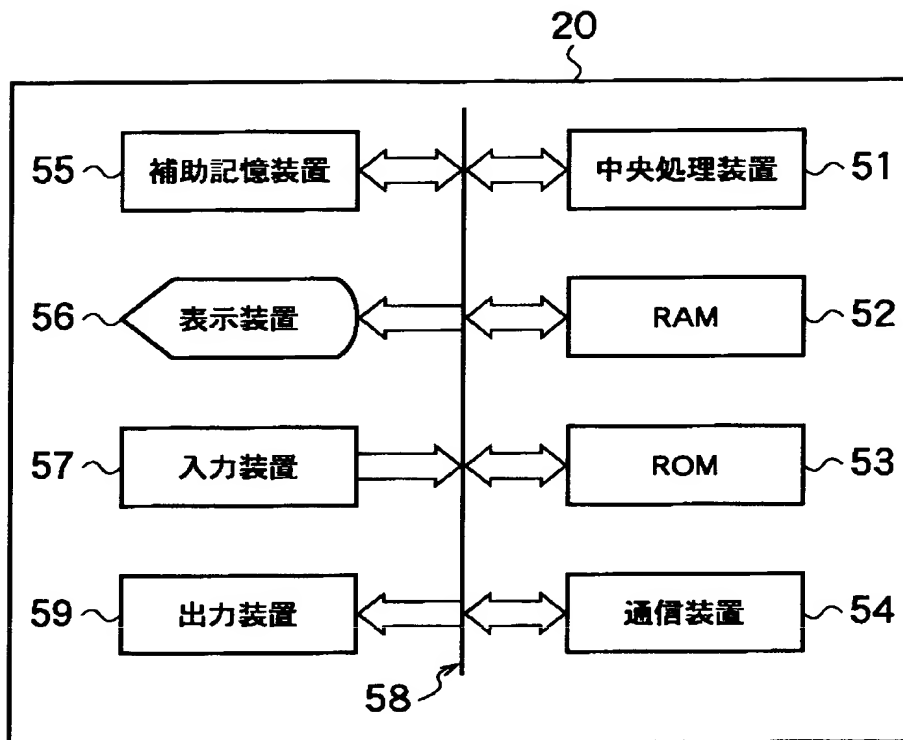
<判定結果データ2>

① N	② イベントの最後に実行状態のタスク	③ ext_tsk
1	taskA	ext_tsk

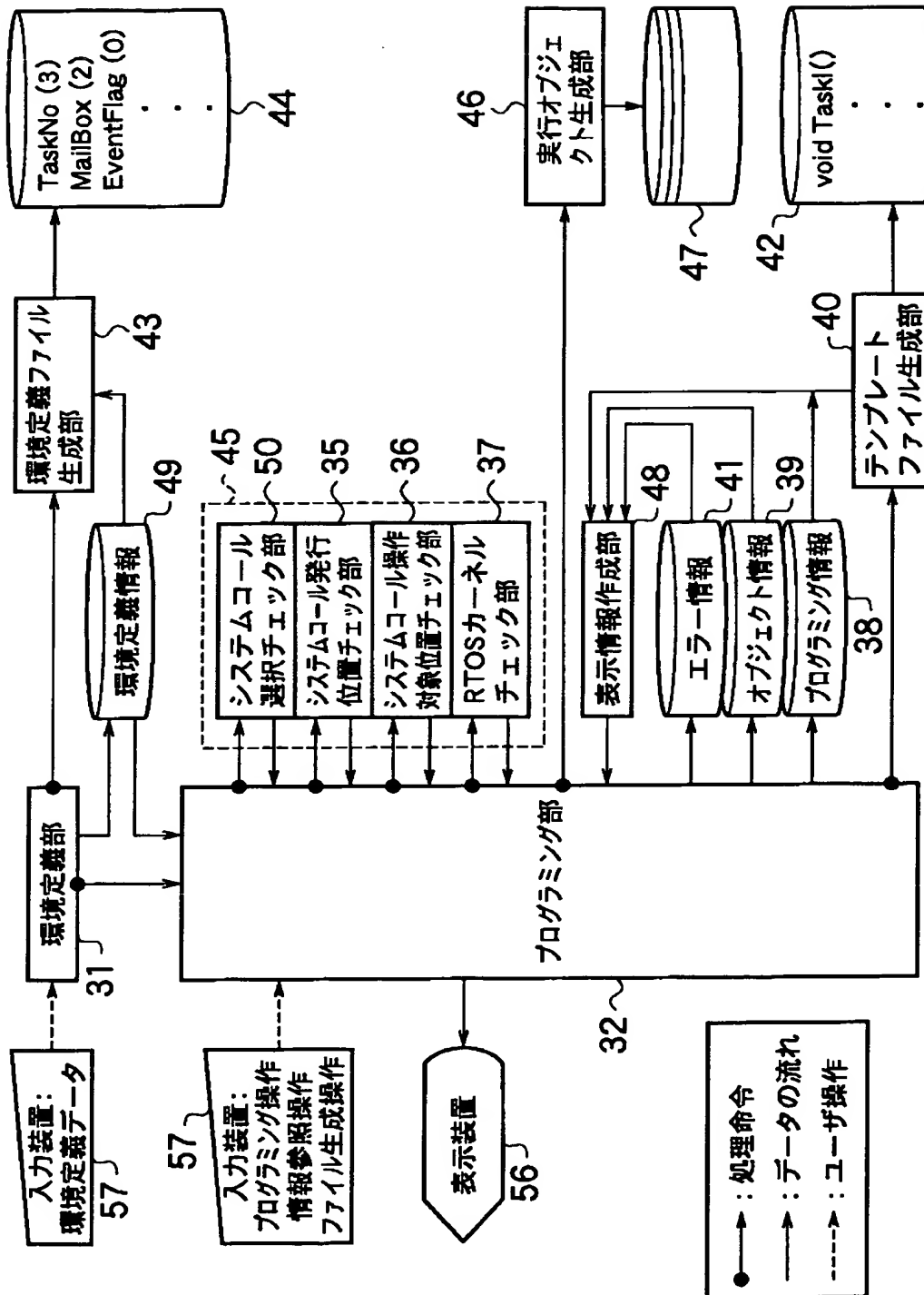
<不具合解決用質問データ>

① N	② 要求タスクを実行可能にするシステムコールの発行対象	③ 要求タスクを実行可能状態にするシステムコール
1	taskA	wup_tsk
2	taskC	sig_sem

【図 2 7】

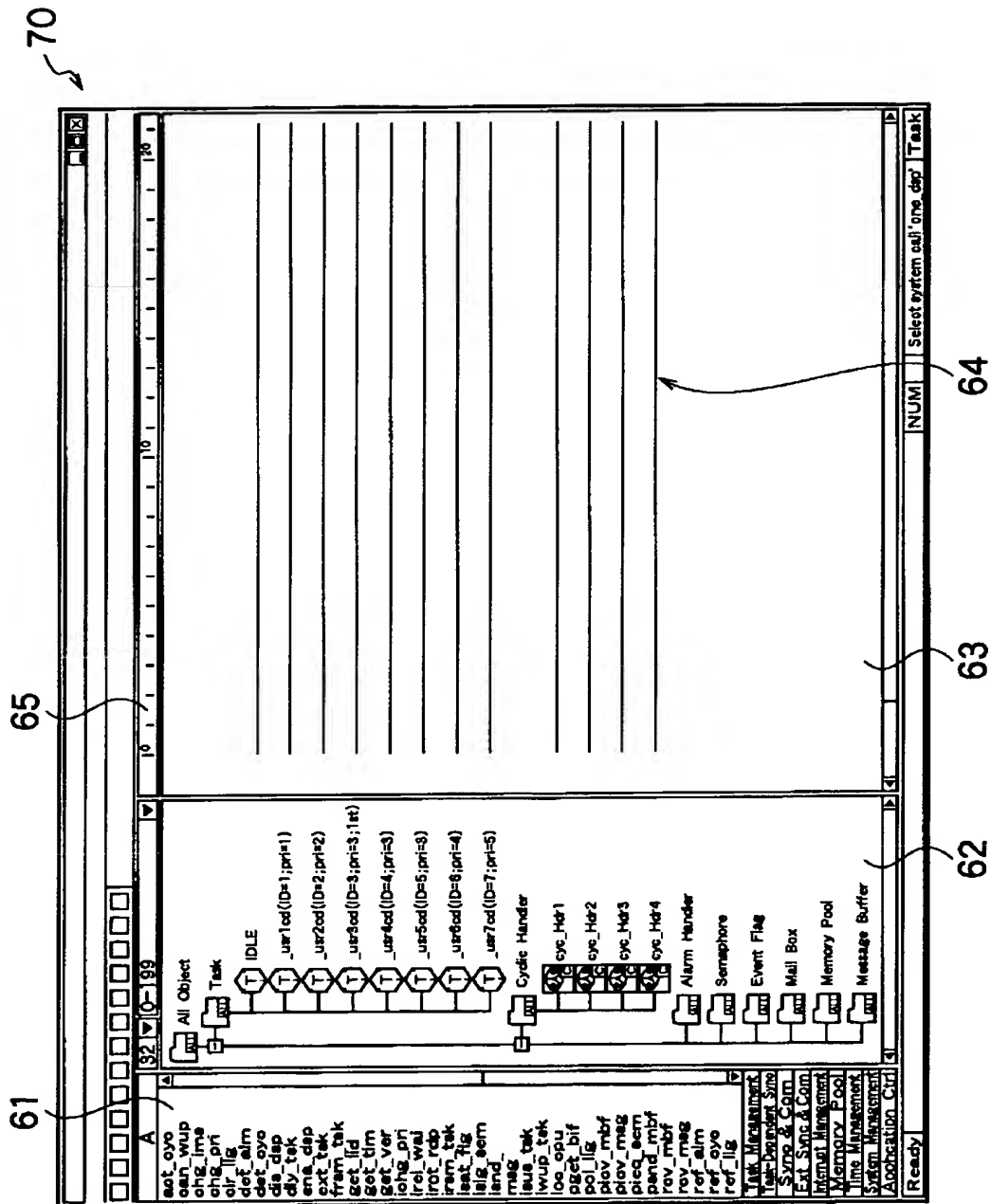


【図 28】

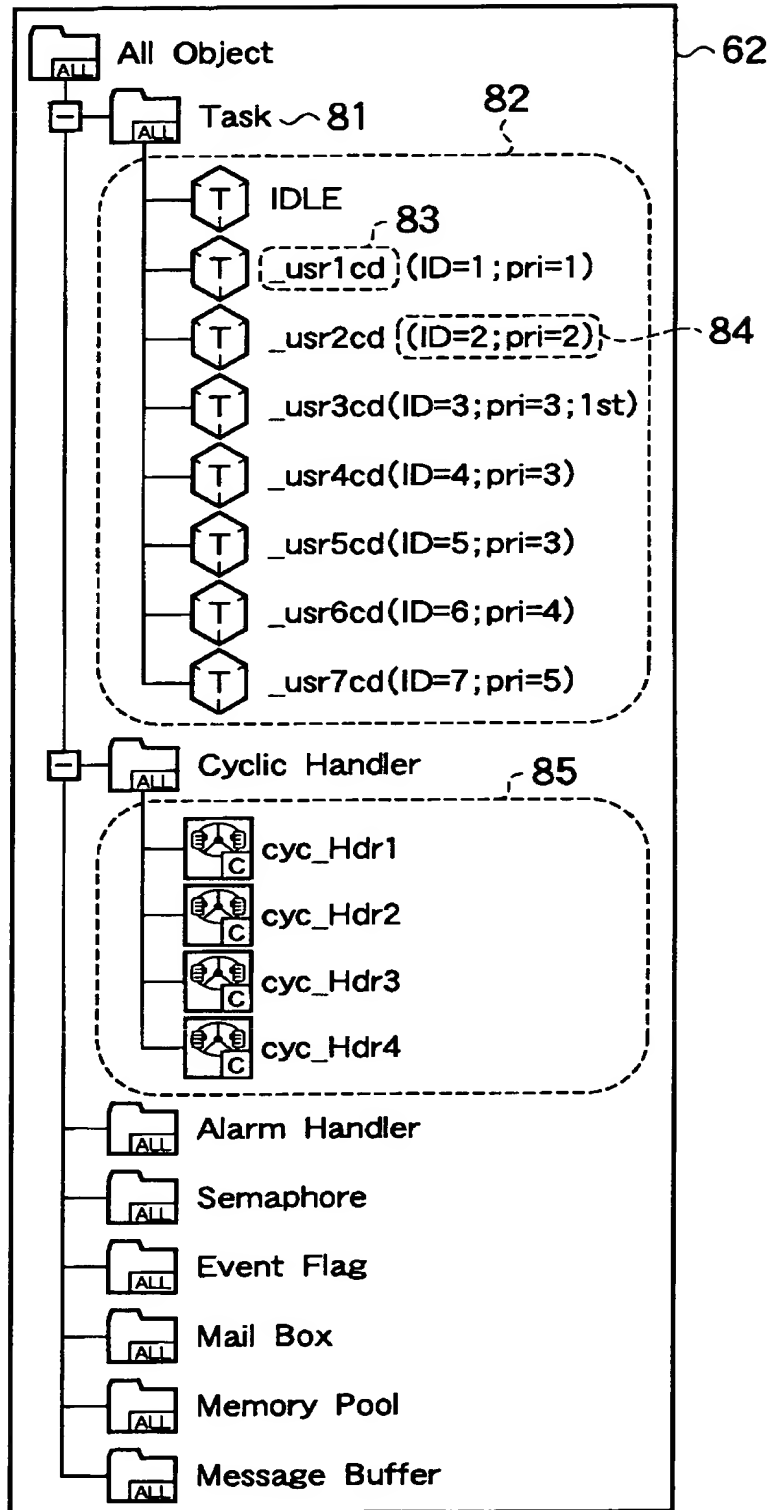




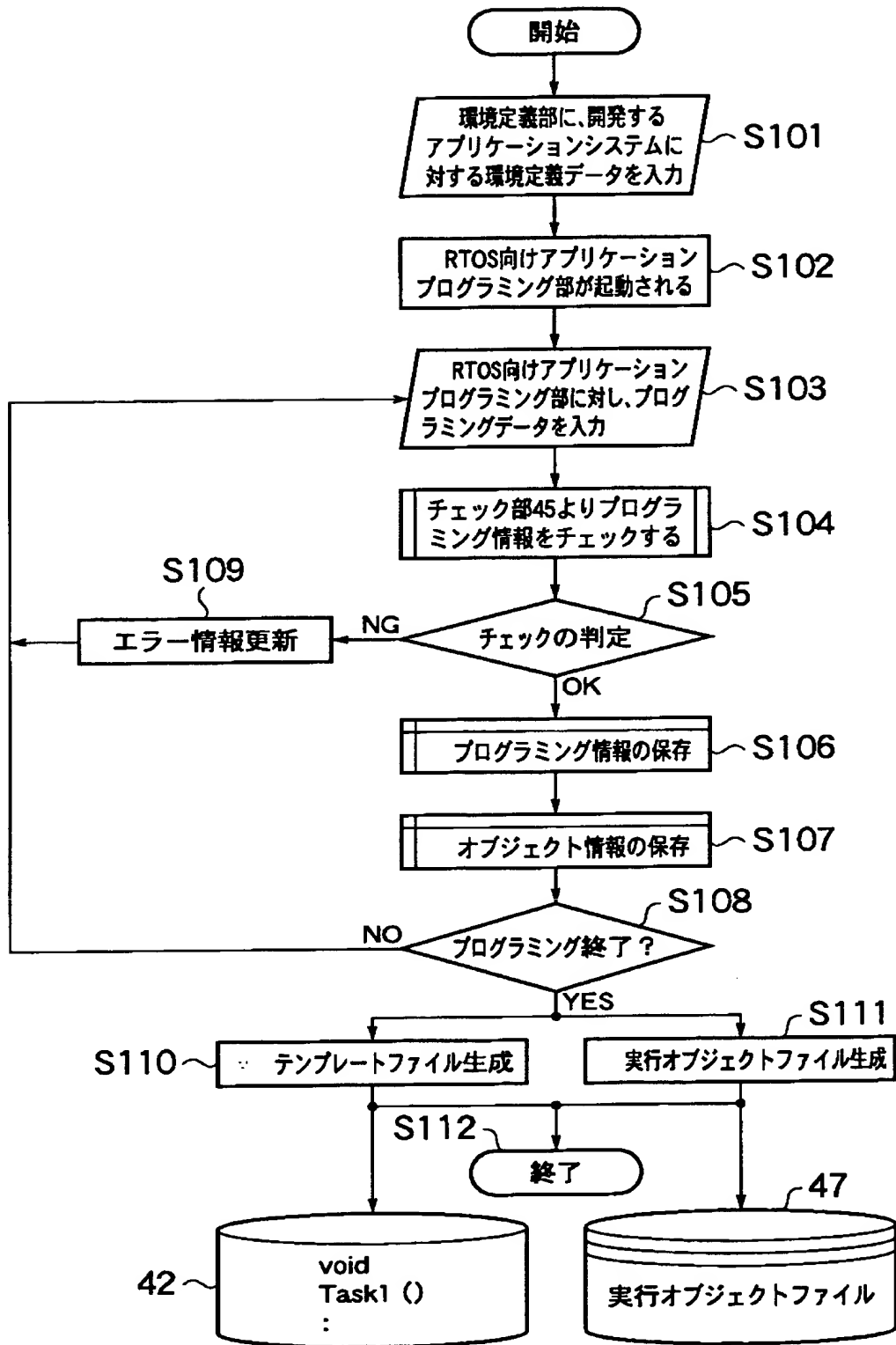
【図 29】



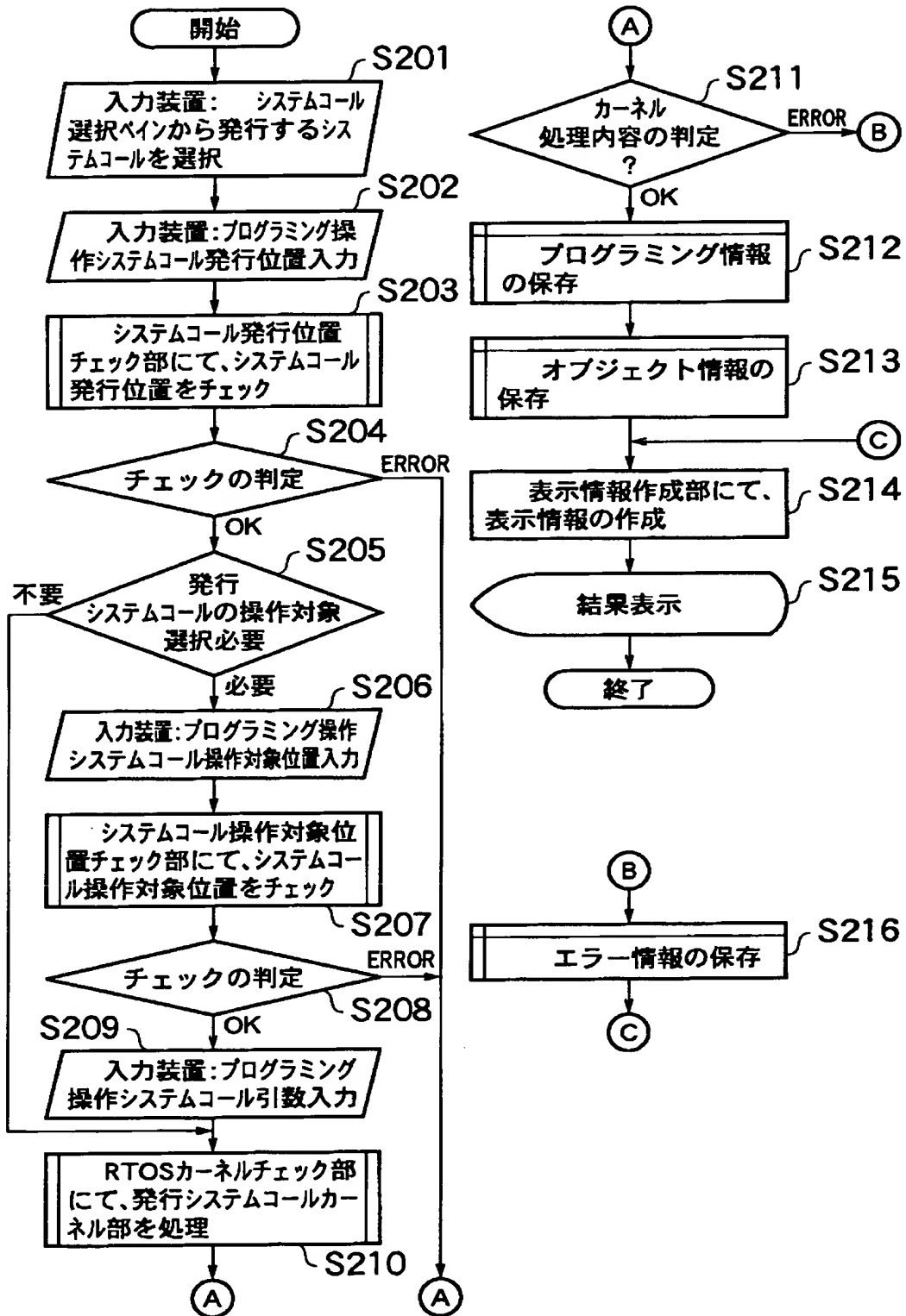
【図 3 0】



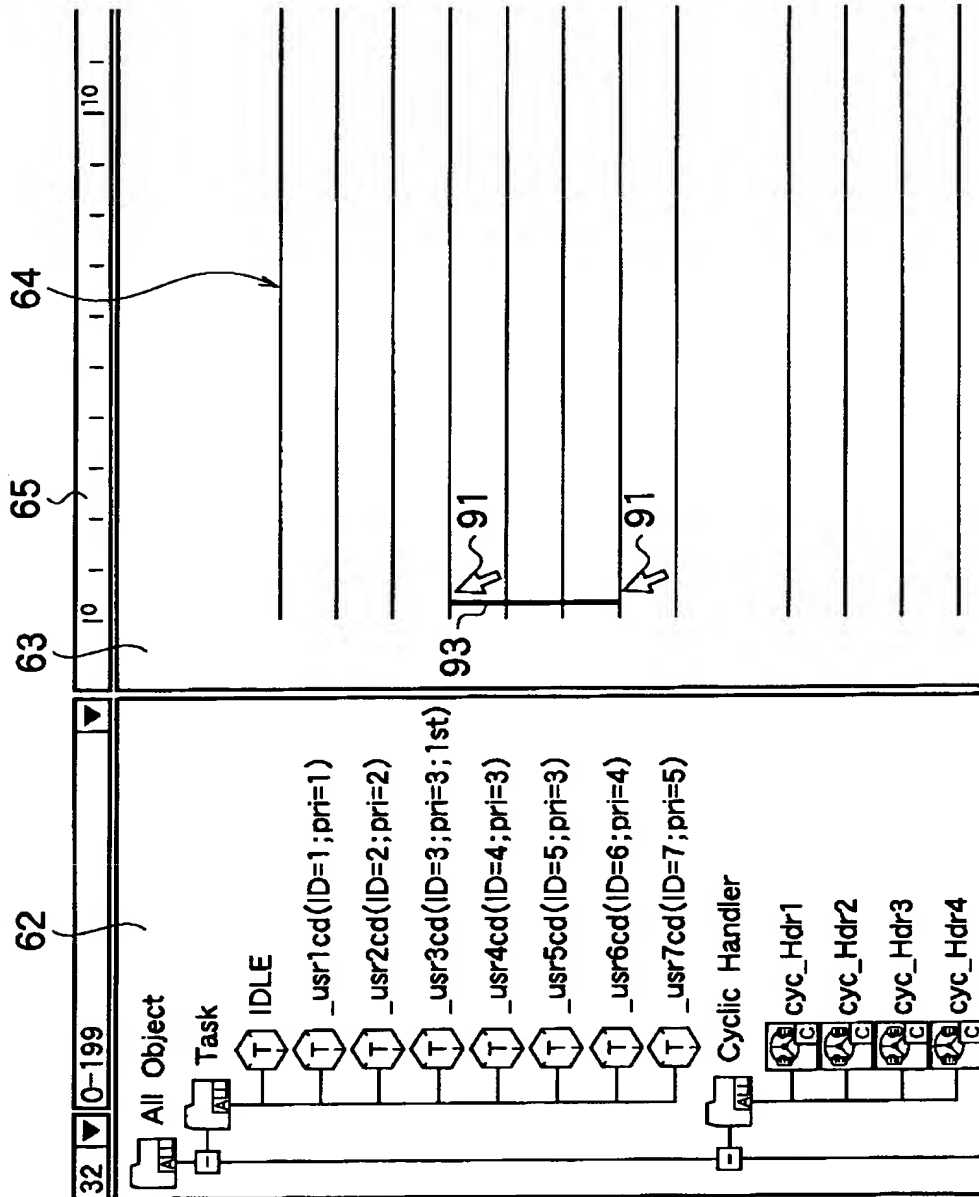
【図 31】



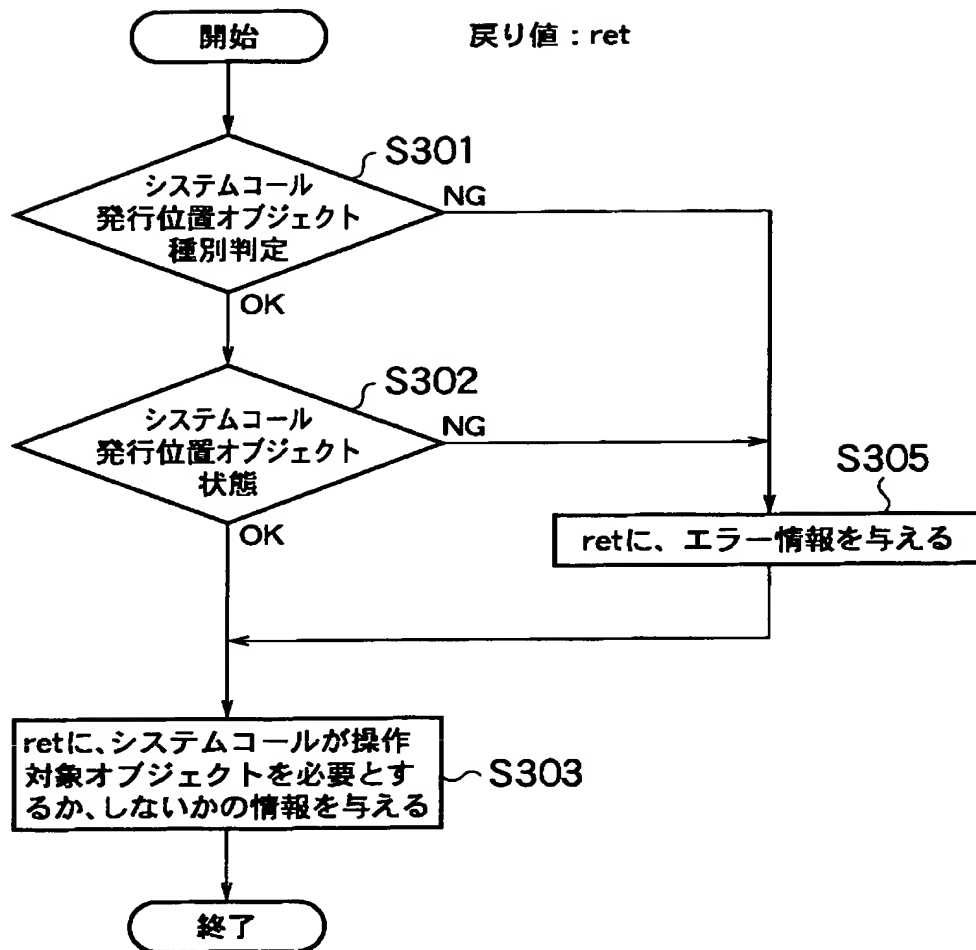
【図 3 2】



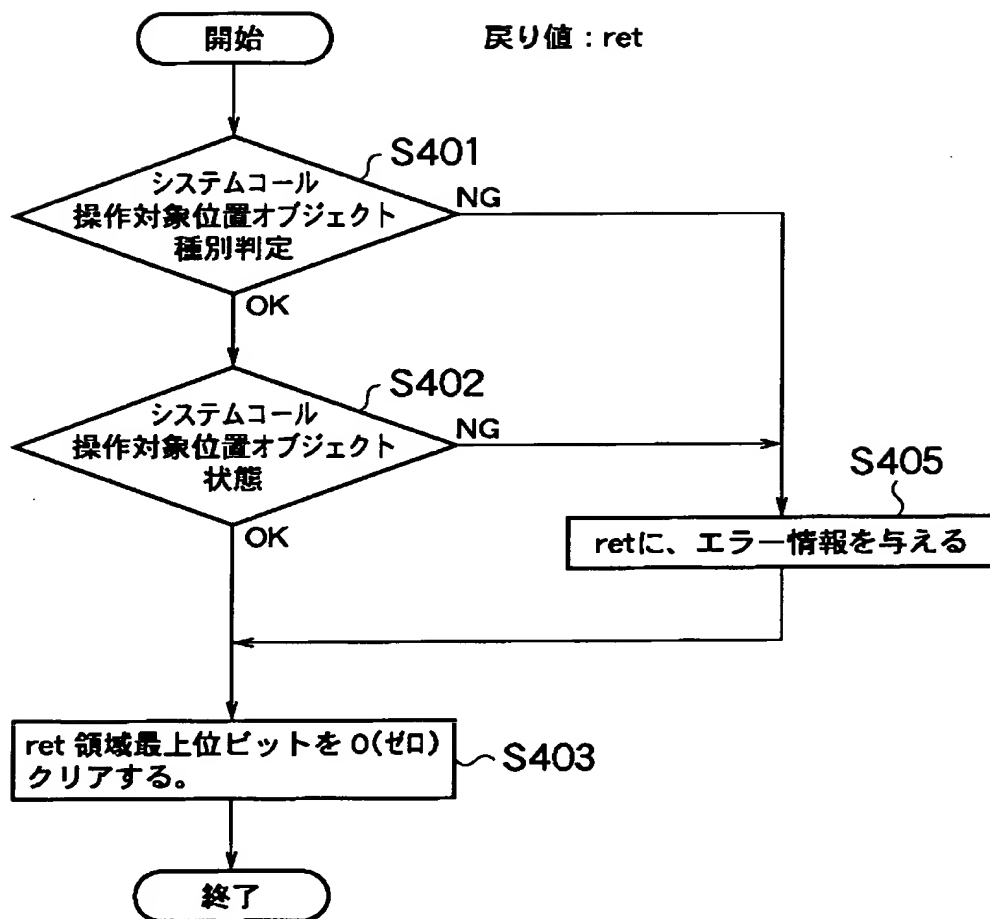
【図 33】



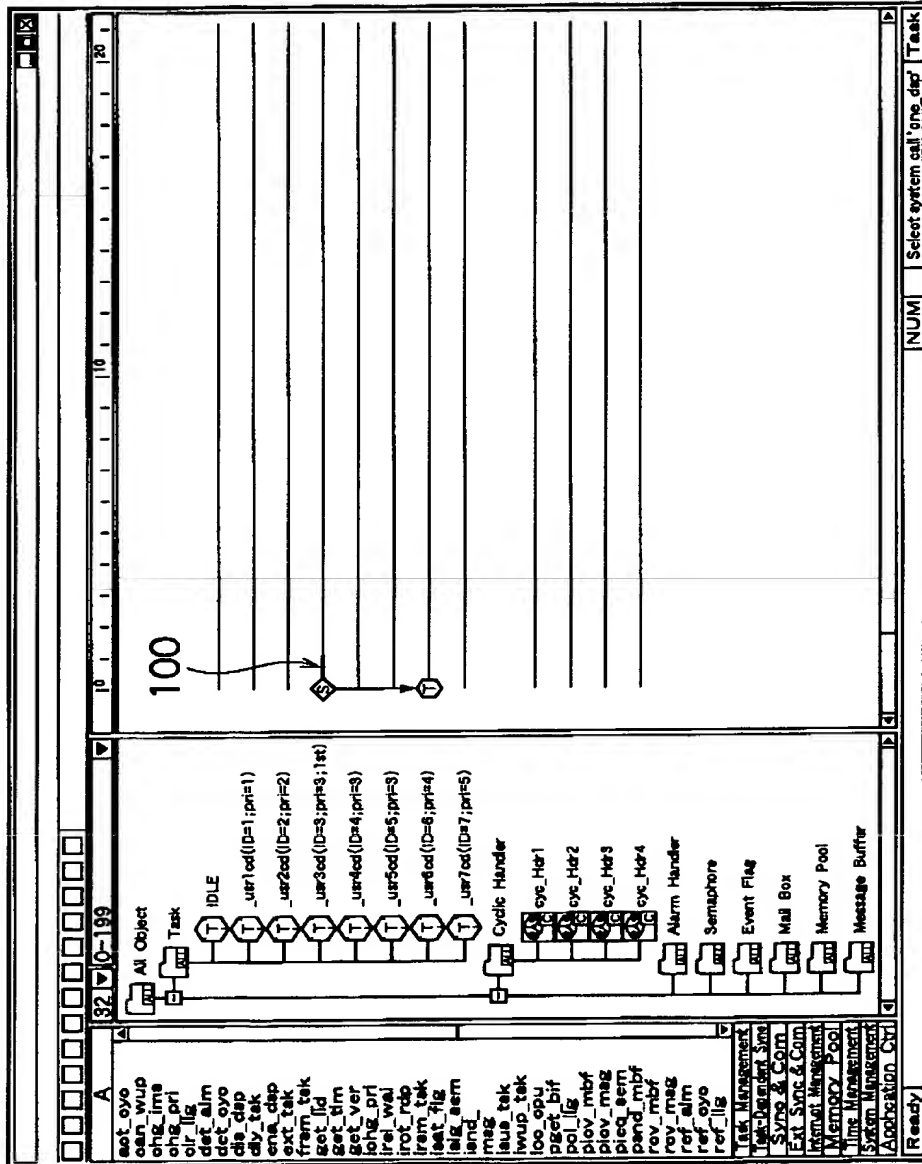
【図 3 4】



【図 3 5】

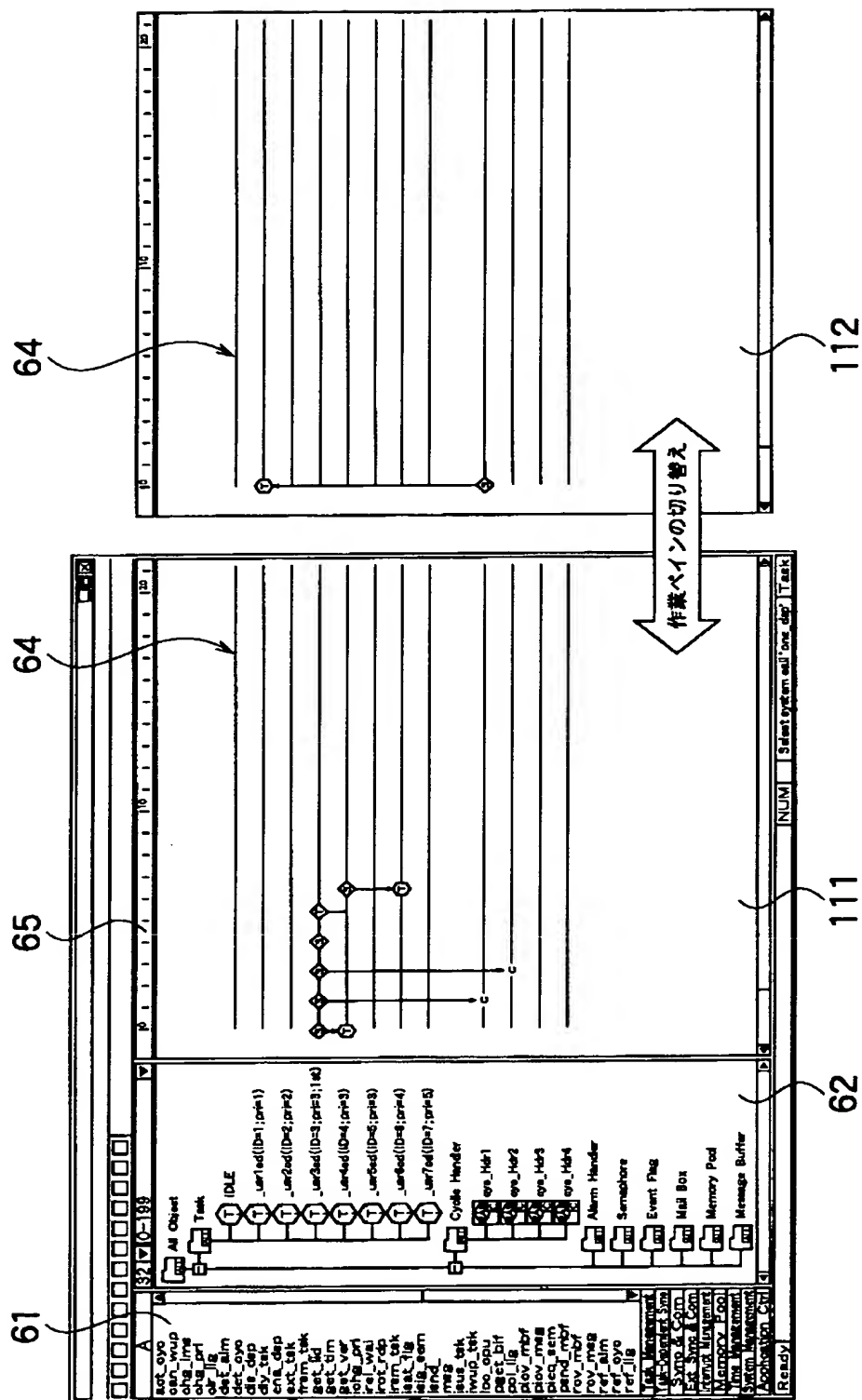


【図 36】

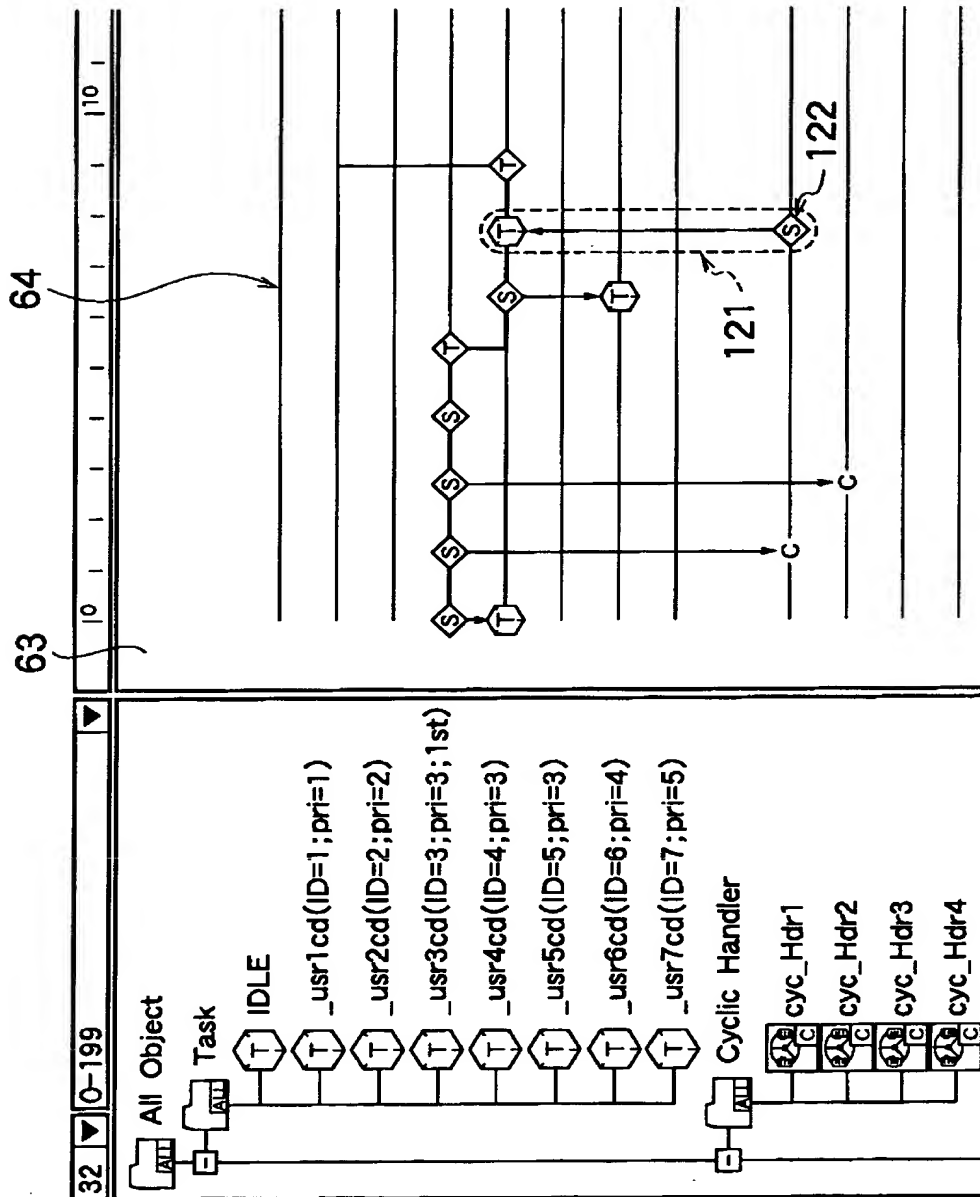




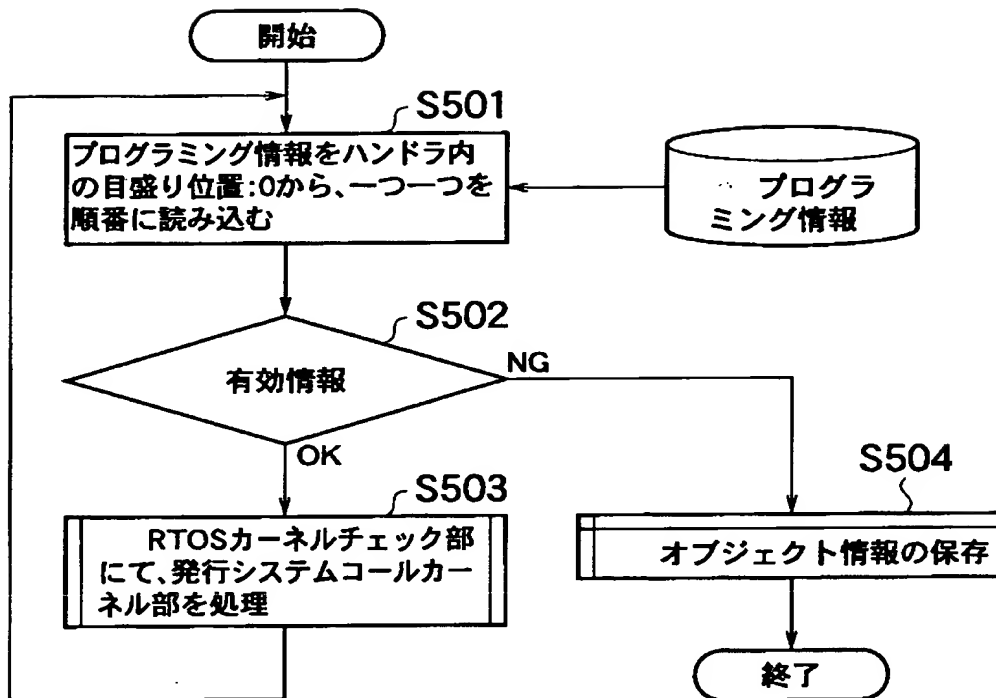
【图 3-7】



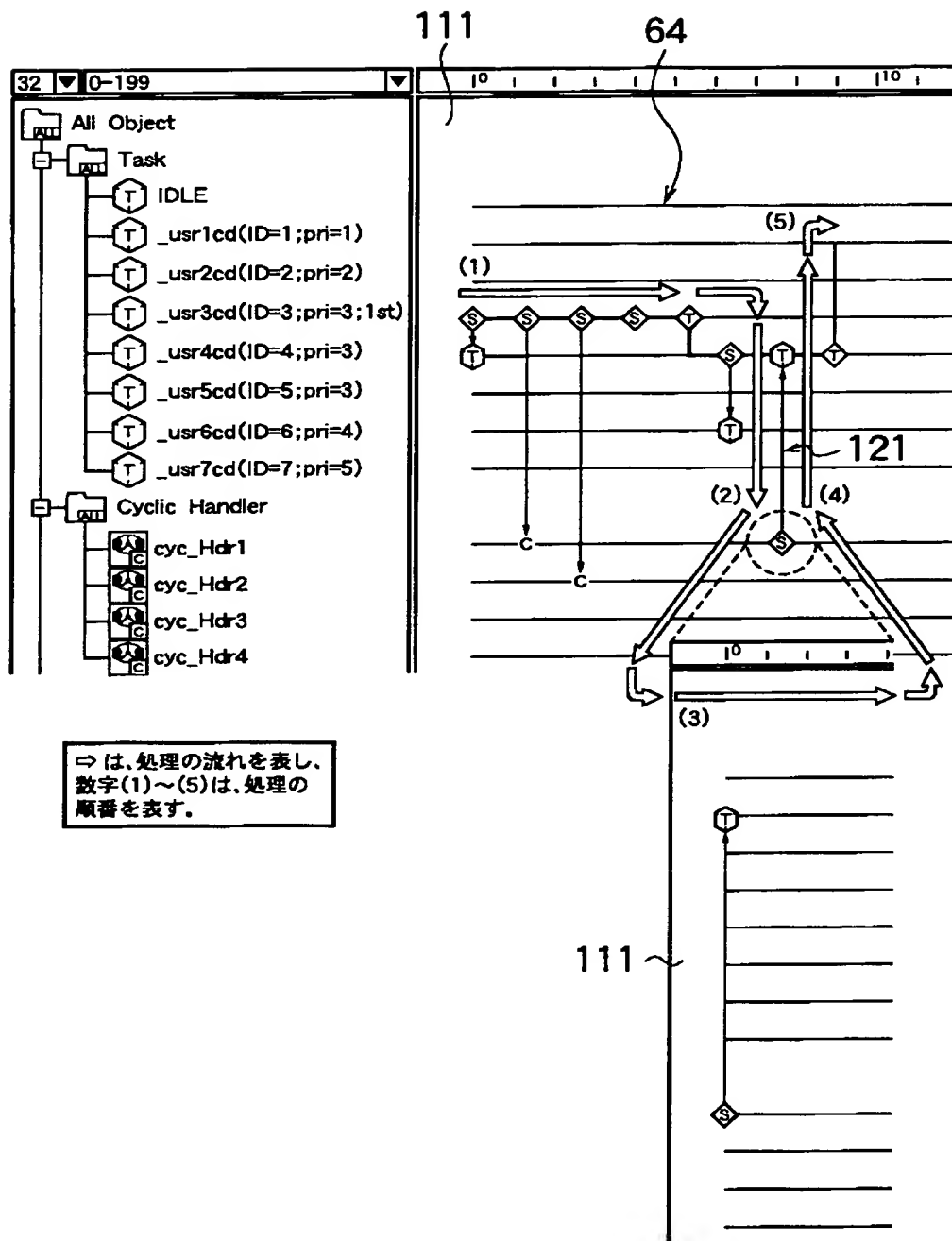
【図 38】



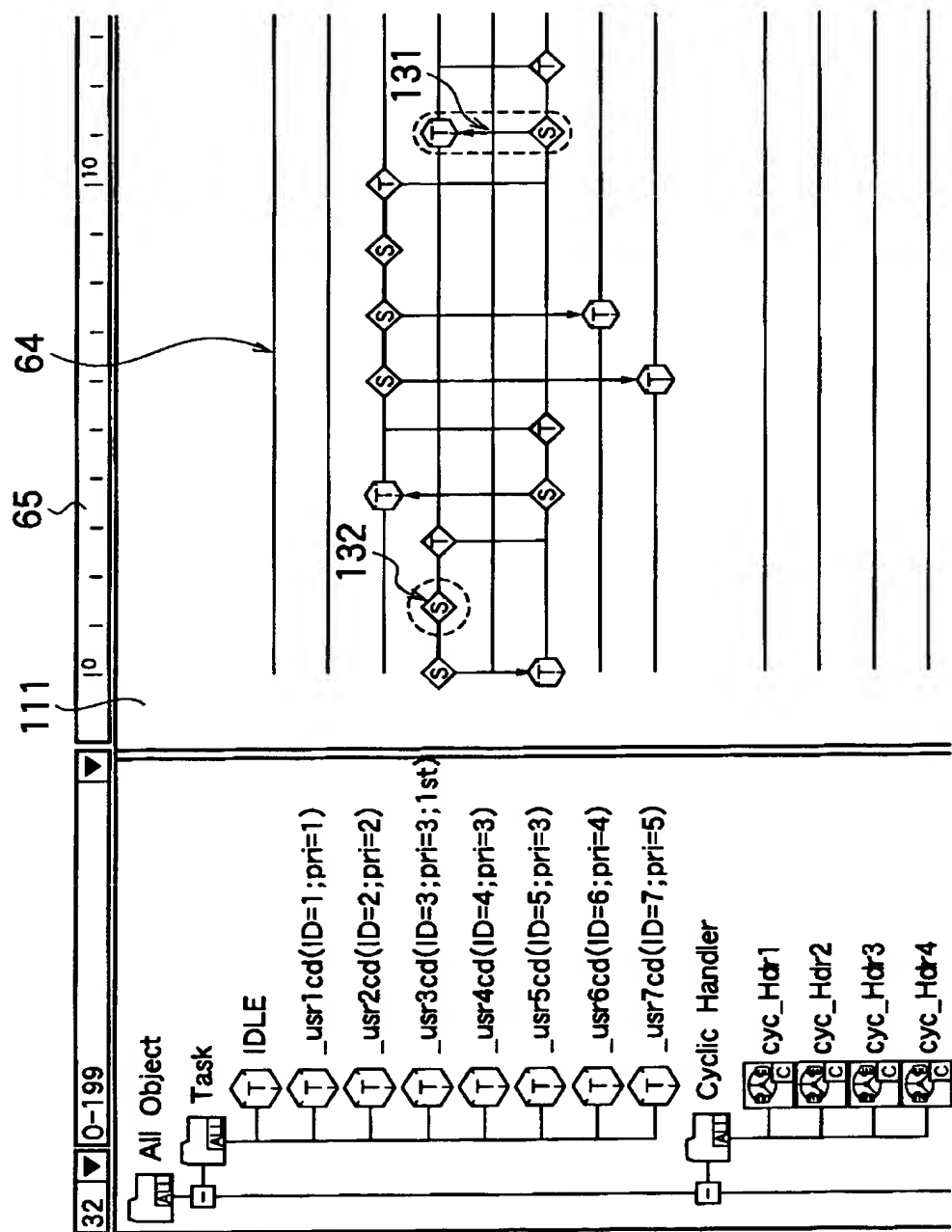
【図 39】



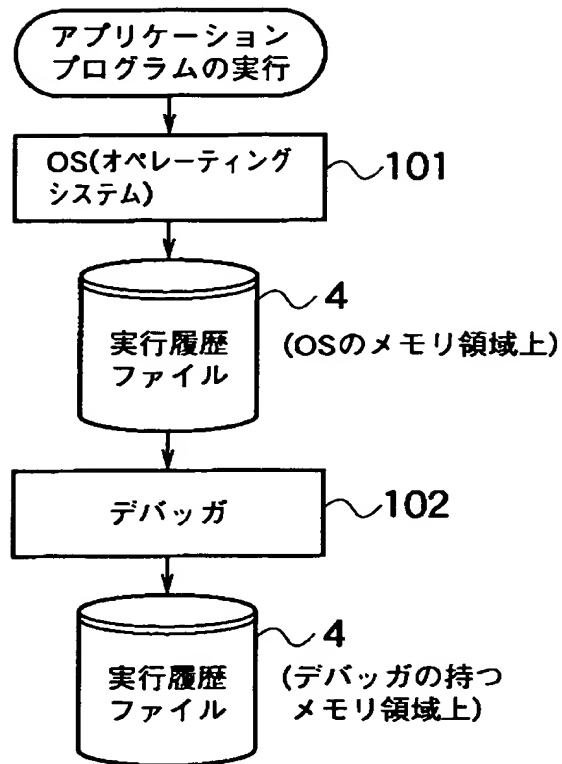
【図 40】



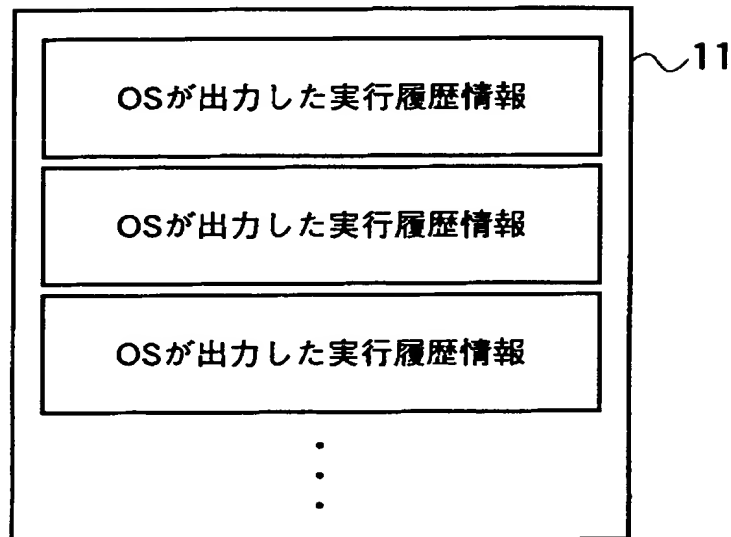
【図 4 1】



【図 4 2】



【図 4 3】



【図 4 4】

(a)

type	oid	sysid	obj
------	-----	-------	-----

(b)

6	0	1		…(イ)
1	1	-9	2	…(ロ)
1	1	-9	3	…(ハ)
1	1	-17		…(ニ)
6	1	3		…(ホ)
1	3	-19	1	…(ヘ)
6	3	1		…(ト)
	⋮			

但し、  
 sysid : sta\_txt . . . . -9  
       : ext\_txt . . . . -10  
       : slp\_txt . . . . -17  
       : wup\_txt . . . . -19  
 とする。

【書類名】 要約書

【要約】

【課題】 対話形式でユーザーから指摘された不具合箇所とプログラムの実行履歴情報から不具合要因と解決策を特定し、これをユーザーに提示し、また、特定した不具合要因と解決策から、該不具合を解決したプログラムを生成してユーザーに提示する。

【解決手段】 プログラムの実行履歴情報をもとに、該プログラムの動作状況を時系列にユーザーに表示する表示手段と、表示された動作状況の中でユーザーが不具合の箇所を指定するための入力手段と、前記入力手段によってユーザーから指定された不具合の箇所と該プログラムの動作状況から該不具合要因を解析し、この不具合要因を解決するための解決策を特定する動作解析手段と、を有し、前記動作解析手段は、特定した解決策を反映させた前記動作状況を再作成し、前記表示手段は、前記不具合要因と前記解決策と再作成した動作状況とをユーザーに表示する。

【選択図】 図 1



認定・付加情報

特許出願の番号	特願 2 0 0 0 - 3 9 6 1 1 2
受付番号	5 0 0 0 1 6 8 4 3 6 9
書類名	特許願
担当官	濱谷 よし子 1 6 1 4
作成日	平成 1 3 年 1 月 1 2 日

<認定情報・付加情報>

【特許出願人】

【識別番号】	598010562
【住所又は居所】	神奈川県川崎市幸区堀川町 5 8 0 番地
【氏名又は名称】	東芝エルエスアイシステムサポート株式会社

【特許出願人】

【識別番号】	000003078
【住所又は居所】	神奈川県川崎市幸区堀川町 7 2 番地
【氏名又は名称】	株式会社東芝

【代理人】

申請人	
【識別番号】	100083806
【住所又は居所】	東京都港区虎ノ門 1 丁目 2 番 3 号 虎ノ門第一ビル 9 階 三好内外国特許事務所
【氏名又は名称】	三好 秀和

【選任した代理人】

【識別番号】	100068342
【住所又は居所】	東京都港区虎ノ門 1 丁目 2 番 3 号 虎ノ門第一ビル 9 階 三好内外国特許事務所
【氏名又は名称】	三好 保男

【選任した代理人】

【識別番号】	100100712
【住所又は居所】	東京都港区虎ノ門 1 丁目 2 番 3 号 虎ノ門第一ビル 9 階 三好内外国特許事務所
【氏名又は名称】	岩▲崎▼ 幸邦

【選任した代理人】

【識別番号】	100100929
【住所又は居所】	東京都港区虎ノ門 1 丁目 2 番 3 号 虎ノ門第一ビル 9 階 三好内外国特許事務所
【氏名又は名称】	川又 澄雄

次頁有

認定・付加情報（続き）

【選任した代理人】

【識別番号】 100108707

【住所又は居所】 東京都港区虎ノ門1丁目2番3号 虎ノ門第1ビル9階三好内外国特許事務所

【氏名又は名称】 中村 友之

【選任した代理人】

【識別番号】 100095500

【住所又は居所】 東京都港区虎ノ門1丁目2番3号 虎ノ門第一ビル9階 三好内外国特許事務所

【氏名又は名称】 伊藤 正和

【選任した代理人】

【識別番号】 100101247

【住所又は居所】 東京都港区虎ノ門1丁目2番3号 虎ノ門第一ビル9階 三好内外国特許事務所

【氏名又は名称】 高橋 俊一

【選任した代理人】

【識別番号】 100098327

【住所又は居所】 東京都港区虎ノ門1丁目2番3号 虎ノ門第一ビル9階 三好内外国特許事務所

【氏名又は名称】 高松 俊雄

出 願 人 履 歴 情 報

識別番号 [ 5 9 8 0 1 0 5 6 2 ]

1. 変更年月日	1 9 9 8 年 1 月 2 3 日
[変更理由]	新規登録
住 所	神奈川県川崎市幸区堀川町 5 8 0 番地
氏 名	東芝エルエスアイシステムサポート株式会社

出 願 人 履 歴 情 報

識別番号 [ 0 0 0 0 0 3 0 7 8 ]

1. 変更年月日	1 9 9 0 年 8 月 2 2 日
[変更理由]	新規登録
住 所	神奈川県川崎市幸区堀川町 7 2 番地
氏 名	株式会社東芝